NASA CR-62021

PROGRAM DOCUMENTATION FOR

A SELECTIVE DISSEMINATION SYSTEM FOR

NASA SCIENTIFIC AND TECHNICAL INFORMATION

JUNE 1966

Prepared under Contract NASw-695 by International Business Machines Corp., Yorktown Heights, New York for the National Aeronautics and Space Administration

Cover

ABSTRACT

Record formats, operating instructions, and flow charts are presented for a selective dissemination of information (SDI) program requiring an IBM 7090/94 computer with 32K storage, two IBM 7607 data channels, and at least eight IBM 729 tape drives. Three component programs are documented: (1) vocabulary control (VOCON), which updates the dictionary and user interest profiles and edits document indexes and user profiles against the dictionary; (2) profile matching (MATCH), which compares document indexes with user profiles and generates announcements when match criteria are met; and (3) user response posting (POST). The programs run under modified versions of the Fortran II Monitor System and in general are written in FAP. Development of the SDI program and its test operations are described in NASA CR-62020 and NASA TM X-57001.

CR-62021

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $3.75

NASA PREFACE

Distribution of selected information to specific individuals is becoming of increasing importance as scientific and engineering staffs grow in number and as the volume of the literature makes it increasingly difficult and time consuming to keep abreast of current advances. In 1963, NASA contracted with the International Business Machines Corporation Advanced Systems Development Division for the design, programming, and test operation of a large scale developmental selective dissemination of information (SDI) system. This contractual program terminated in December 1964. The program phases undertaken during this period were preliminary operations in a continuing NASA developmental SDI effort.

This report presents record formats, operating instructions, and flow charts for the particular SDI system operated for NASA during the contractual development of the program. As documented in this report, the operation requires an IBM 7090 or 7094 computer with 32K core storage, two 7607 data channels, and eight IBM 729 tape units. The programs run under modified versions of the Fortran II Monitor System and in general are written in FAP.

An associated report, NASA CR-62020, Implementation, Test and Evaluation of a Selective Dissemination of Information System for NASA Scientific and Technical Information, issued June 1966, gives details of the program operation up to December 1964. It describes the preparation of user interest profiles, and illustrates the abstract cards and response cards received by system participants. It also includes statistics on system performance, and presents the results of a questionnaire concerning user opinions and comments.

These reports should be read in conjunction with a NASA report, NASA TM-X-57001, NASA Selective Dissemination of Information Program (IBM 7090/94 System), issued June 1966. The latter presents operating experience with the program after completion of the preliminary phases and transfer of the program to operation by the NASA Scientific and Technical Information Facility. It also describes program modification and discusses the availability of the 7090/94 computer program.

Operation of the particular SDI program described in these three reports was discontinued by NASA in February 1966. Its termination was the result of continuing evaluation and evolution in all areas of the NASA SDI program. The 7090/94 program was replaced by an IBM 1410 system. The change recognized continuing advances in dissemination techniques and computer technology. A bibliographic search program written for an IBM 1410 with 40K memory and process overlap and priority features was modified for SDI operations and was

demonstrated to work effectively. An IBM Systems/360 Model 40 computer scheduled for installation by the NASA Scientific and Technical Information Facility during the summer of 1966 will include emulator hardware for 1410 operation during transition of SDI operations to the new computer system.

In addition to conversion of computer operations to a different computer, the form of announcement was changed early in 1966. Users of the NASA SDI service now receive a computer-printed listing of citations rather than the abstract cards. The current NASA SDI program thus differs substantially from that described in this report and in NASA TM-X-57001 and NASA CR-62020. These reports are published as a record of a unique SDI system and a stage in the development of selective dissemination of information systems.

The IBM 7090/94 SDI program will be made available on request to organizations interested in studying this SDI system. The source program, documentation, and associated off-line IBM 1401 and 1410 programs described in report NASA TM-X-57001 can be supplied by special arrangement with NASA. Program maintenance would be the responsibility of the organization receiving it and no guarantee concerning its operation can be made.

Further information concerning the NASA SDI programs may be obtained from:

Scientific and Technical Information Division
Code USD
National Aeronautics and Space Administration
Washington, D.C. 20546

CR 62021

NASA CR-62021

PROGRAM DOCUMENTATION FOR

A SELECTIVE DISSEMINATION SYSTEM FOR

NASA SCIENTIFIC AND TECHNICAL INFORMATION

JUNE 1966

Prepared under Contract NASw-695 by International Business Machines Corp., Yorktown Heights, New York for the National Aeronautics and Space Administration

APSTRACT

Record formats, operating instructions, and flow charts are presented for a selective dissemination of information (SDI) program requiring an IBM 7090/94 computer with 32K storage, two IBM 7607 data channels, and at least eight IBM 729 tape drives. Three component programs are documented: (1) vocabulary control (VOCON), which updates the dictionary and user interest profiles and edits document indexes and user profiles against the dictionary; (2) profile matching (MATCH), which compares document indexes with user profiles and generates announcements when match criteria are met; and (3) user response posting (POST). The programs run under modified versions of the Fortran II Monitor System and in general are written in FAP. Development of the SDI program and its test operations are described in NASA CR-62020 and NASA TM X-57001.

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $3.75

NASA PREFACE

Distribution of selected information to specific individuals is becoming of increasing importance as scientific and engineering staffs grow in number and as the volume of the literature makes it increasingly difficult and time consuming to keep abreast of current advances. In 1963, NASA contracted with the International Business Machines Corporation Advanced Systems Development Division for the design, programming, and test operation of a large scale developmental selective dissemination of information (SDI) system. This contractual program terminated in December 1964. The program phases undertaken during this period were preliminary operations in a continuing NASA developmental SDI effort.

This report presents record formats, operating instructions, and flow charts for the particular SDI system operated for NASA during the contractual development of the program. As documented in this report, the operation requires an IBM 7090 or 7094 computer with 32K core storage, two 7607 data channels, and eight IBM 729 tape units. The programs run under modified versions of the Fortran II Monitor System and in general are written in FAP.

An associated report, NASA CR-62020, Implementation, Test and Evaluation of a Selective Dissemination of Information System for NASA Scientific and Technical Information, issued June 1966, gives details of the program operation up to December 1964. It describes the preparation of user interest profiles, and illustrates the abstract cards and response cards received by system participants. It also includes statistics on system performance, and presents the results of a questionnaire concerning user opinions and comments.

These reports should be read in conjunction with a NASA report, NASA TM-X-57001, NASA Selective Dissemination of Information Program (IBM 7090/94 System), issued June 1966. The latter presents operating experience with the program after completion of the preliminary phases and transfer of the program to operation by the NASA Scientific and Technical Information Facility. It also describes program modification and discusses the availability of the 7090/94 computer program.

Operation of the particular SDI program described in these three reports was discontinued by NASA in February 1966. Its termination was the result of continuing evaluation and evolution in all areas of the NASA SDI program. The 7090/94 program was replaced by an IBM 1410 system. The change recognized continuing advances in dissemination techniques and computer technology. A bibliographic search program written for an IBM 1410 with 40K memory and process overlap and priority features was modified for SDI operations and was

demonstrated to work effectively. An IBM Systems/360 Model 40 computer scheduled for installation by the NASA Scientific and Technical Information Facility during the summer of 1966 will include emulator hardware for 1410 operation during transition of SDI operations to the new computer system.

In addition to conversion of computer operations to a different computer, the form of announcement was changed early in 1966. Users of the NASA SDI service now receive a computer-printed listing of citations rather than the abstract cards. The current NASA SDI program thus differs substantially from that described in this report and in NASA TM-X-57001 and NASA CR-62020. These reports are published as a record of a unique SDI system and a stage in the development of selective dissemination of information systems.

The IBM 7090/94 SDI program will be made available on request to organizations interested in studying this SDI system. The source program, documentation, and associated off-line IBM 1401 and 1410 programs described in report NASA TM-X-57001 can be supplied by special arrangement with NASA. Program maintenance would be the responsibility of the organization receiving it and no guarantee concerning its operation can be made.

Further information concerning the NASA SDI programs may be obtained from:

Scientific and Technical Information Division
Code USD
National Aeronautics and Space Administration
Washington, D.C. 20546

NASA-SDI SYSTEM

CONTENTS:

This volume is solely devoted to the documentation of the programs for the NASA-SDI System.

A. SYSTEM DESCRIPTION, NASA-SDI

SDI - the Selective Dissemination of Information - implements the philosophy that the supplying of relevant information to persons desiring it, from the bulk of information available, can be accomplished effectively and economically by machine.

SDI may be said to be the reverse of a conventional library operation. A library stores documents and waits for users to come in the door. An SDI system stores representations of users' interests, and, as documents come in the door, the system disseminates to users notices of documents that appear relevant to their interests.

An SDI user is represented in the system by a profile i.e., a list of descriptors that delineates his fields of interest. Information, the content of a document, is similarly represented. Periodically, the system edits user profile changes, new user profiles, and incoming document profiles against a vocabulary control guide, to insure that all profiles utilize the same descriptors to identify the same things. Then, the system compares these profiles with each other, selecting the documents apparently relevant to the stated interests of each user. For any comparison, matches on given numbers of descriptors determine relevancy. Notices of the documents are then sent to users. A notice consists of the document abstract and descriptors printed on a 3 x 5 card and of a Port-A-Punch® card with which copies of the document may be ordered and actual degree of relevance indicated.

Two major computer programs comprise the SDI system designed for the National Aeronautics and Space Administration, called NASA-SDI. These two programs are: the Vocabulary Control Program (VOCON), and the Profile Matching Program (MATCH). VOCON updates or originates the vocabulary control guide, updates or originates the user profiles, and edits document and user profiles against the vocabulary control guide. MATCH compares document and user profiles, generating document notices when the designated match criteria are met. In sections B and C following, VOCON and MATCH are presented from a programming and operating standpoint, including detailed record formats and program flow charts. They are shown schematically in Figures 1 and 2. For a condensed description of what is accomplished by VOCON and MATCH in terms of the SDI System phase, refer to Volume I of this report, Section B (NASA-CR-62020).

A third major computer program (POST), and its preprocessor (NPOST), are also presented in sections D and E following. They are shown schematically in Figure 3. POST is the SDI Posting Program. It maintains the historical notice-response file, recording or posting user responses to the notices in the file and preparing second notices to accompany documents requested by users. The purpose of the historical notice-response file is the accumulation of statistical information about system performance and the handling of document requests. NPOST loads on tape the user responses (returned notices) and POST control cards, checks the responses for validity, and limits the number of responses sent to POST at any one time.

POST and NPOST are not completely integrated into the NASA-SDI System. NPOST will load NASA-SDI responses on tape, and POST will post them to the historical notice-response file. But, the second notices prepared by POST would require further processing to add NASA addresses to them. Similarly, the notices generated by MATCH that form the skeleton of the historical notice-response file require a change in format before POST can use them. To circumvent the first discrepency, the NASA-SDI System utilizes second notices prepared ahead in quantity with the document number and date omitted. This information is added later when a specific document has been requested. To eliminate the second discrepency, IBM has supplied to NASA a 1401 program that changes the format of the MATCH notices for POST.

To summarize: Three programs are submitted to NASA and presented here as parts of an SDI System. Two of these, VOCON and MATCH, form the necessary parts of the NASA-SDI System. They were prepared by IBM to meet the specific needs of NASA. The third program, POST, was developed in another context. It is a useful adjunct to the NASA-SDI System, but, as submitted, is not integrated completely into the system.

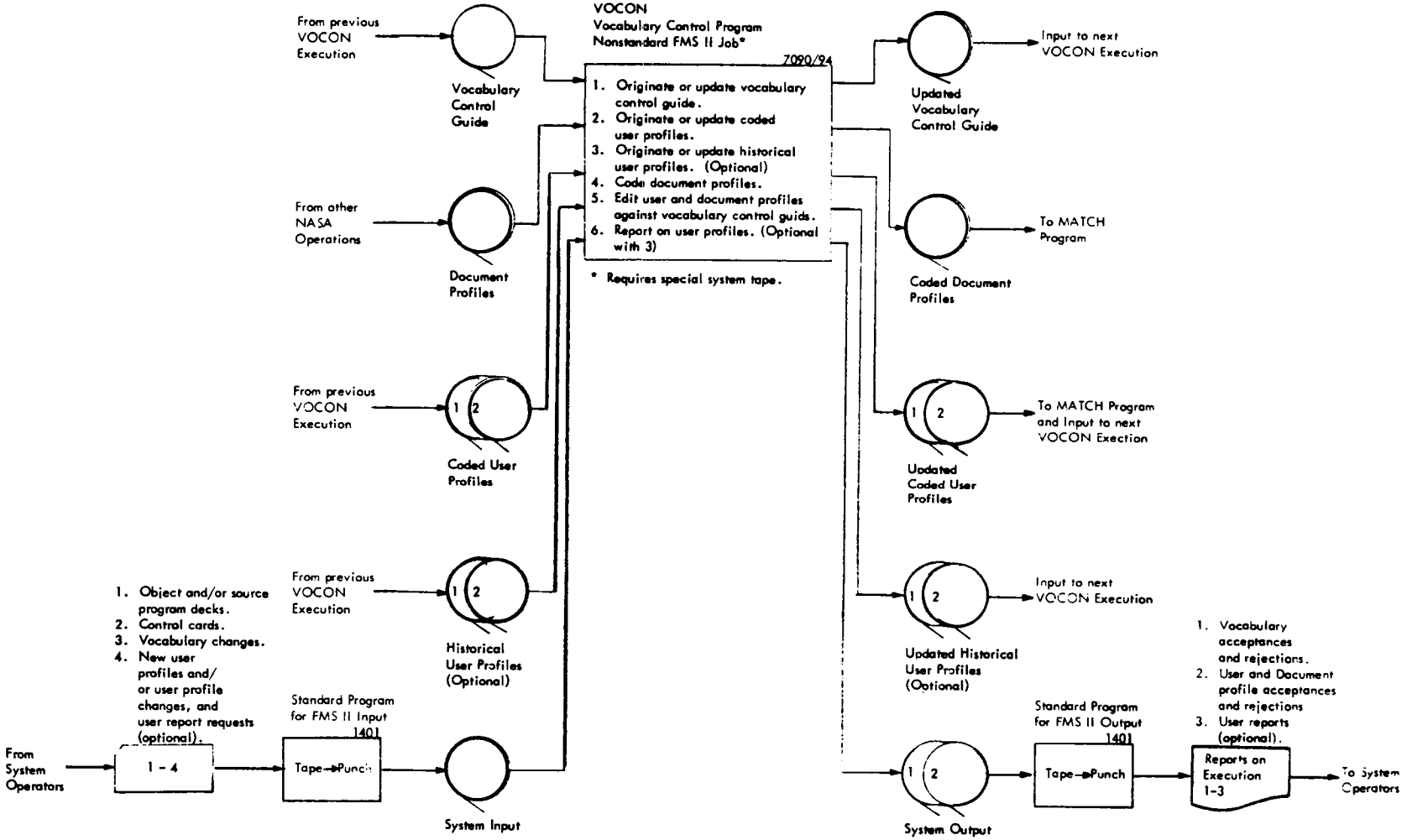


Figure 1. VOCON

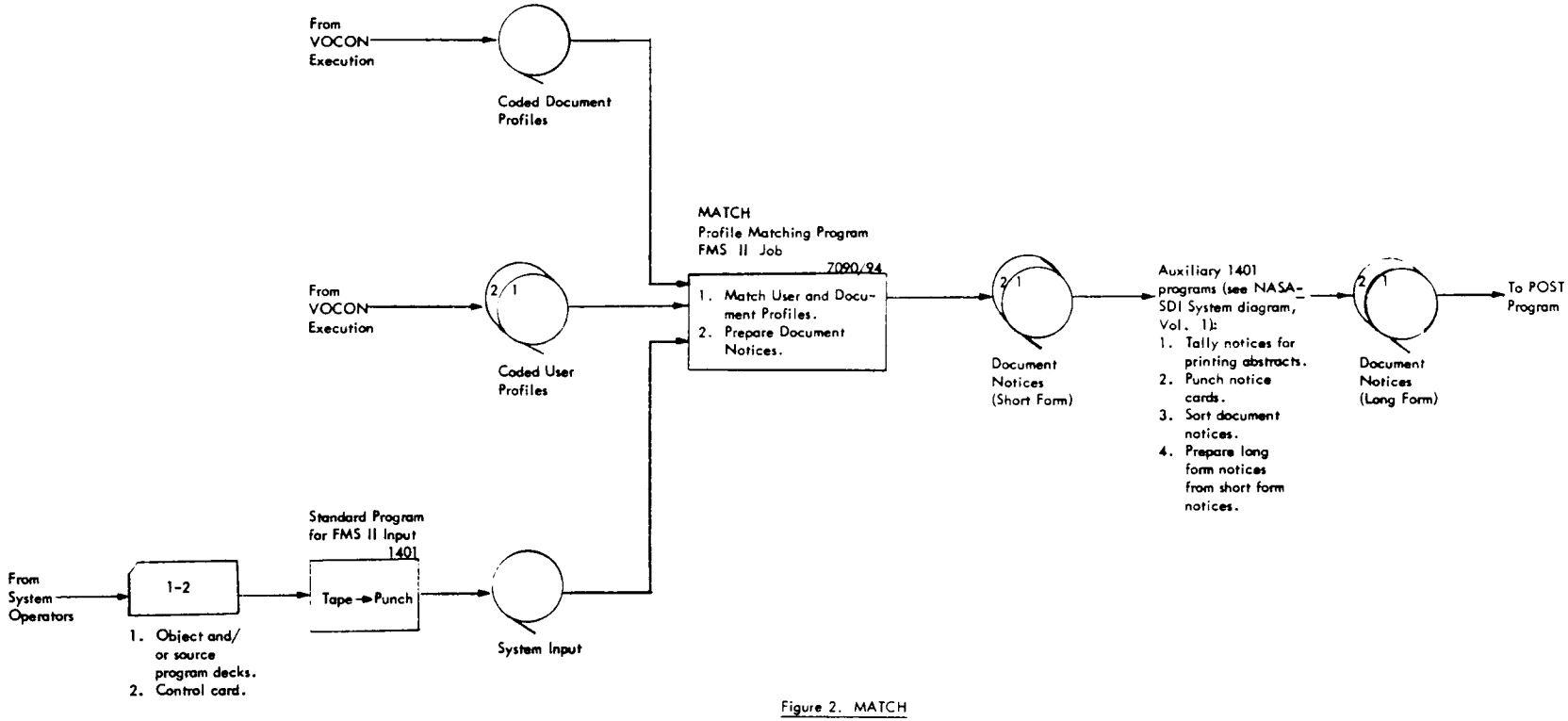


Figure 2. MATCH

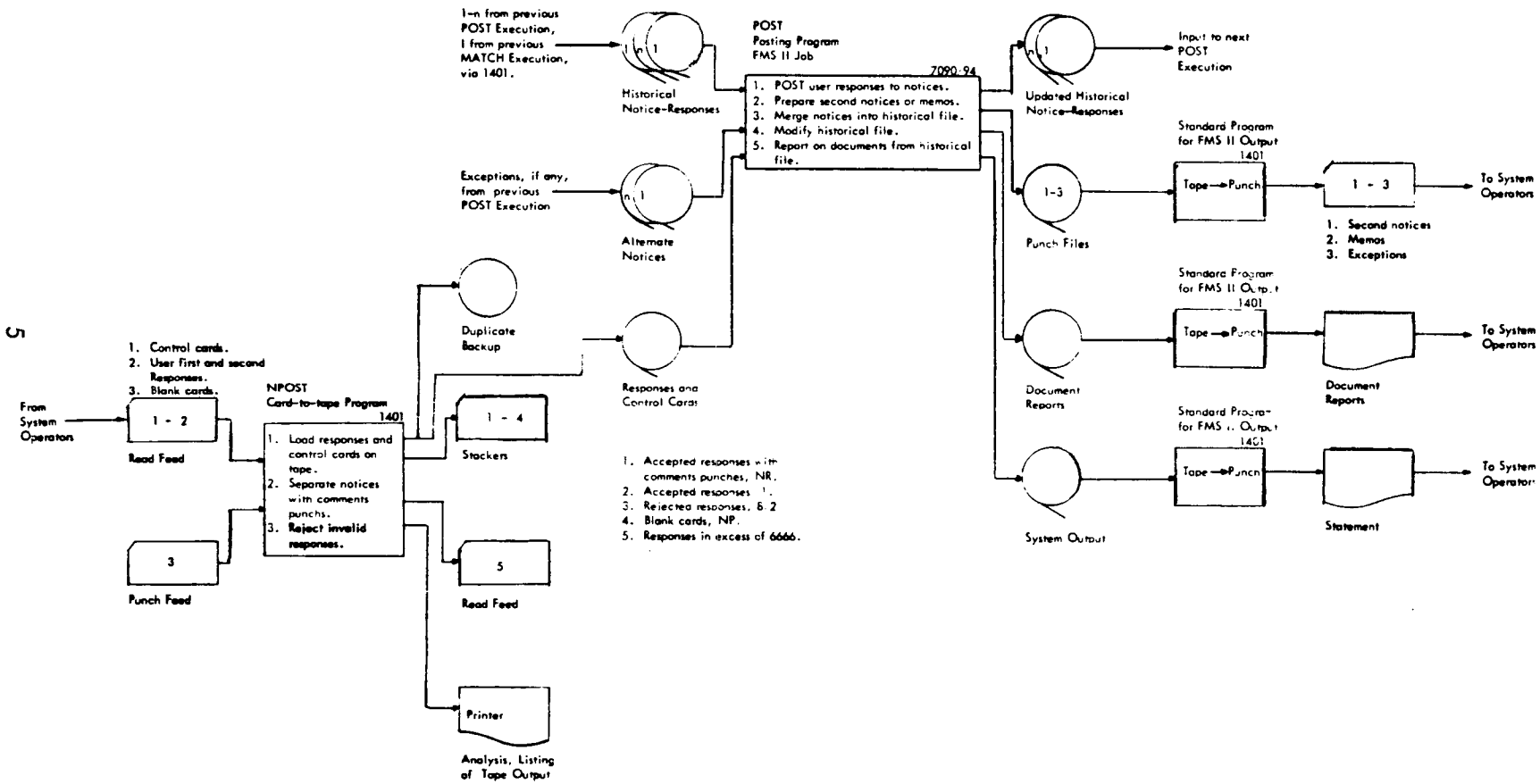


Figure 3. NPOST and POST

B. SDI VOCABULARY CONTROL PROGRAM

1. Program Summary

1.1 Description

The SDI Vocabulary Control Program (VOCON), for the IBM 7090/94 Data Processing System, maintains the vocabulary control guide and prepares and edits user and document profiles for matching. The program runs under a modified version of the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with eight 729 tape units (in addition to FORTRAN units). VOCON is written in FAP and utilizes the FORTRAN input/output package, IOP. It consists of a primary program and several subroutines.

The overall purpose of VOCON, in terms of the SDI system, is to insure that the user and document profiles utilize the same descriptors to identify the same things, so that system accuracy and effectiveness are improved. In other words, descriptors to be used in matching are either accepted or rejected, depending upon whether or not they appear in the vocabulary; but, the vocabulary itself is also accessible to change, as fields of interest appear, expand and contract. Specifically, VOCON updates or originates the vocabulary control guide, updates or originates the user profiles and edits document and user profiles against the vocabulary control guide.

Profile descriptors entering VOCON consist of single words or phrases; user descriptors may also be modified by 'must' or 'not' (for more information on user descriptors, see MATCH). All descriptors in the vocabulary control guide are single words, condensed or coded into single-machine-word values, and the guide is maintained in coded-descriptor order; the English descriptor is also associated with the code. The guide contains primary descriptors that are admissable words (unmodified), secondary descriptors that are synonyms equated to primary descriptors, and trouble terms that are inadmissable words, such as "system". The vocabulary may be changed at any time via word input. The document and user profiles prepared for matching contain only coded descriptors. Optionally, the user profiles may also be maintained in a historical format that includes the English descriptors. The input user profiles enter on cards; the input document profiles are provided on tape via other NASA operations. Phrases input in document profiles are divided into their component words so that the final document profiles are lists of coded single words. These document words also build the vocabulary; any new terms occurring in the documents are automatically added to the vocabulary. Phrases input in the user profiles are maintained as are 'must' and 'not' modifiers, and the component words of phrases are also added to the user profile as single words. User words are not automatically added to the vocabulary.

The processing accomplished by VOCON is divided into nine phases, under the direction of the primary program, MAIN. One or more phases may be dropped from a given execution, depending upon the input to the execution. The purposes of the phases are:

1. Read and process all control cards. Form IO tape usage table. Any error here will cause an error exit with appropriate comment.
2. Read all new input, coding descriptors and forming records on tape A for sorting: vocabulary changes, document profiles, user profile changes, new user profiles. Controlling subroutines GDNRY, GTDOC, GTUSE, CODER.
3. Sort tape A on coded value of descriptors, maintaining profile number (zero if vocabulary change), and item serial number. That is, form an inverted file from tape A. Controlling subroutine SORT.
4. Pass tape A against the vocabulary, simultaneously updating the vocabulary and writing out accepted items, document descriptor items onto tape B and user descriptor items onto tape C. Any vocabulary EQUATE changes will generate modifications for user profiles on tape C. Controlling subroutine UPDATE.
5. Sort tape B on profile number and item serial number. Controlling subroutine SORT.
6. Sort tape C on profile number and item serial number. Controlling subroutine SORT.
7. Create the document profile tape for matching. Controlling subroutine PDTB.
8. Update the user profiles and form the user profile tape for matching. Controlling subroutine UPDUS.
9. Summarize program activity, then exit.

The control cards examined in phase 1 introduce all tape unit assignments and tape number assignments where required, as well as variable program parameters. The presence or absence of control cards and parameters governs the course of the program with respect to creation or updating of various files, etc., or restart. The variable program parameters are:

1.	DATE	Date of computer run
2.	NPHASE	Starting phase for recovery run
3.	MINU	Minimum percentage value of accepted descriptors for retention of user profile
4.	MIND	Minimum percentage value of accepted descriptors for retention of document profile.

1.2 Input

1. Vocabulary control guide
2. Document profiles

3. Coded user profiles
4. Historical user profiles (optional)
5. Control cards, vocabulary changes, new user profiles, user profile changes, user report requests (optional with Item 4).

1.3 Operating Instructions

The SDI VOCON program is a standard FMS system job with a special system tape. The card deck for the system input tape includes in this order: ** job, *ID, * IOP, * XEQ, source decks if any, binary decks if any, * DATA, control cards, vocabulary changes if any, new user profiles if any, user profile changes if any, EOF. The following instructions and comments apply to the three types of runs, initial, update, and restart. Online messages trace the execution and provide operator instructions.

A. Initial Run

Run to create the vocabulary and/or user profile tape(s) and/or document profile tape.

1. At the start of the job the following tapes should be mounted on the drives specified in control card 6:

   a. System input tape
   b. IP document tape (if any)
   c. OP vocabulary tape
   d. OP historical profile tape (if any)
   e. OP user profile tape, reel 1 (if any)
   f. System output tape
   g. For any logical tape number found on card 6, mount a scratch tape on the appropriate drive if none of the tapes mentioned above (a to f) is mounted on it.

2. At the end of phase 4 the OP vocabulary tape will be rewound and unloaded. A message will be printed to mount the second reel of the OP user profile tape or a pool tape if no second reel. The computer will halt until this tape is mounted and START is pressed.

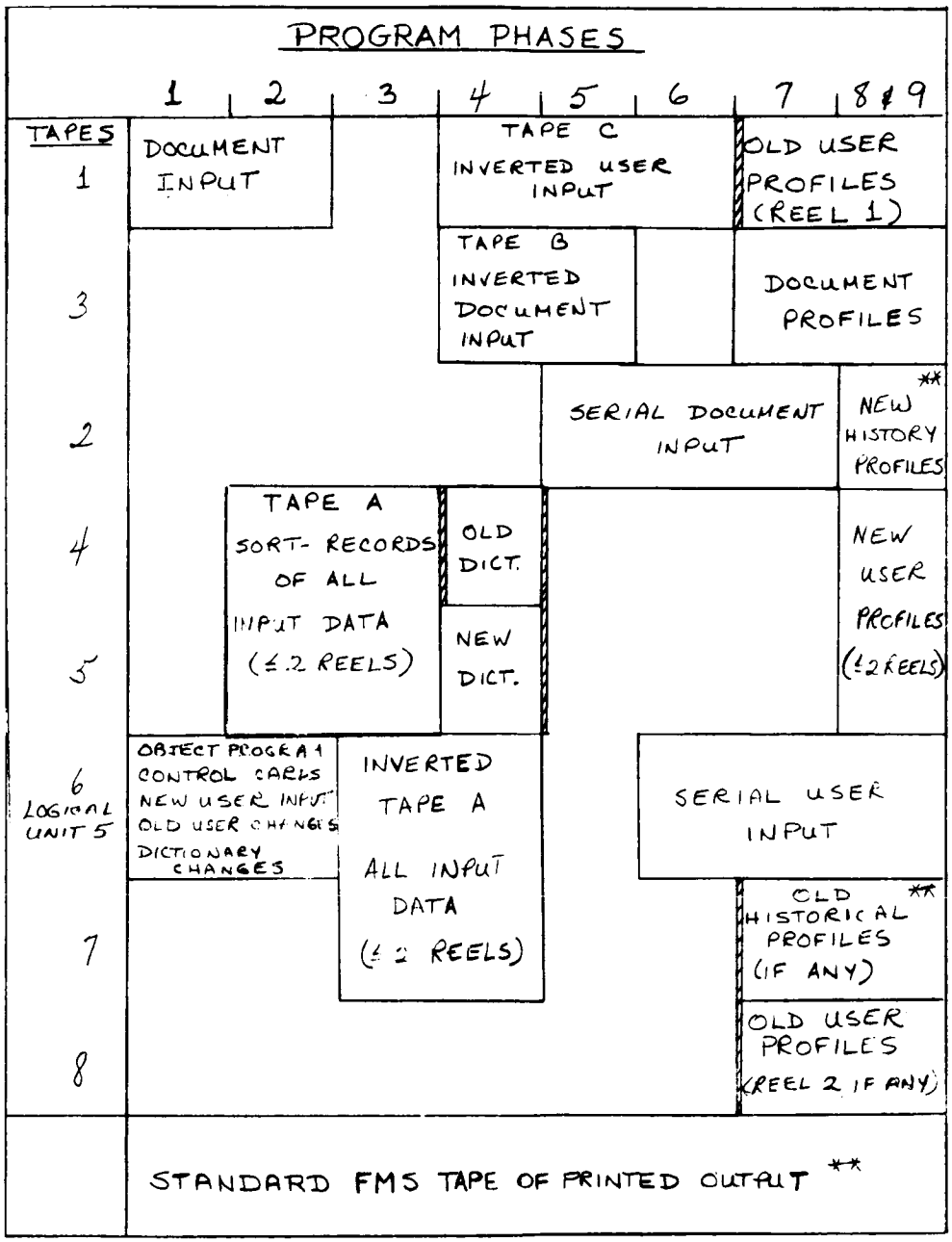
3. At the beginning of phase 7, a message will be printed to mount the IP user profile tape and the first reel of the IP historical user profile tape if these tapes are needed for this run. However, the computer will not stop for this mounting operation.

4. During phase 8, a message to mount the second reel of the IP historical user profile tape will be printed if necessary.

5. During any phase of the program a message to mount a new system output tape will be printed if necessary.

B. Update Run

Run to update the vocabulary tape and/or to create or update the



ODD TAPES → CHANNEL B EVEN TAPES → CHANNEL A IF FORTRAN LOGICAL 5 IS ON CHANNEL A, OTHERWISE WE HAVE THE REVERSE

**LOAD ADDITIONAL TAPES AS REQUIRED

Figure 4. VOCON Tape Usage Scheme

user profile tape and/or to create the document profile tape. The only differences from an initial run are:

1. Instead of reel 1 of the OP user profile tape, a pool tape should be mounted on the specified drive.

2. A message will be printed during phase 3 to mount the IP vocabulary tape on its appropriate drive.

3. At the end of phase 4, both the IP and the OP vocabulary tape will be rewound and unloaded. A message will be printed to mount the two reels of the OP user profile tape. If there are not two reels of OP user profiles the message will indicate that a pool tape is to be mounted.

C. Restart Run

If either an initial run or an update run is terminated before completion of phase 9, the following procedures should be observed:

1. Save all online messages.

2. Remove all tapes noting their respective drives.

3. If the last phase printed online is either phase 1 or phase 2, the job must be rerun from the beginning. A job will be terminated in either of these phases with an online message if a control-card error or a redundant tape exists.

In the first case, the input deck setup should be checked and the missing card inserted or an erroneous card corrected as the error warrants, and the job should then be rerun. In the second case, if the input tape is redundant, a new input tape will have to be generated and the job rerun. If tape A (the tape being written in phase 2) is redundant, a new pool tape should be substituted for that tape, all tapes should be remounted on their respective drives, and the job rerun with the original input tape.

4. If the last phase printed online is phase 3 or beyond:

   a. The IP document profile tape need not be mounted but a pool tape must be mounted in its place.

   b. Vocabulary and/or user cards should be removed from the input deck setup; thus, the last two cards in the deck setup will be control card 8 and the EOF card. It is very important that card 8 contain the additional punches described if there originally was any document and/or user input or vocabulary EQUATE changes. Also, the phase number on card 8 will be the last

phase number appearing in the online messages. These changes, therefore, necessitate the use of a new input tape when restarting any job beyond phase 2.

c. If the job was terminated because an incorrect tape was mounted, a message will be printed indicating this situation. The correct tape should be substituted for this tape and all tapes should then be remounted on their respective drives and the job restarted.

d. If the job was terminated because of a redundant tape during any of the sorting phases (phase 3, phase 5 and phase 6), a new pool tape should be substituted for that tape, all tapes should be remounted on their respective drives and the job restarted using the new input tape.

5. In phases 4, 7 and 8 where a restart job needs a new input tape and also requires the tape generated on the FMS system input tape unit, change control card 6 in cols. 37-38 to the logical tape number for a drive not being used, mount the generated tape on this drive and mount the new input tape on the FMS system input tape unit.

1.4 Output

1. Vocabulary control guide, updated
2. Coded document profiles
3. Coded user profiles, updated
4. Historical user profiles, updated (optional)
5. Printout of vocabulary acceptances and rejections, user and document profile acceptances and rejections, user reports (optional with Item 4).

2. Record Formats

2.1 Input

1. Vocabulary Control Guide

The vocabulary control guide consists of 101 files. File 1 is a label. Files 2-101 are vocabulary records sorted on coded descriptor value.

a. File 1

Record	
1	label of 12 words (72 characters)
2	catalog of entries: 100 consecutive pairs of computer words corresponding to the upper and lower limits of a corresponding data file. The first entry in this catalog will always be a word of all zeros. For files having no entries, both limits will be computer words of all sevens.
3	100 computer words each of which contains the number of logical records in the corresponding file.

b. Files 2, 3, 4, ..., 101

Each of these files contains approximately 1/100th of the entire vocabulary. Each physical record is 450 words long consisting of 15 descriptor records each 30 words in length. Each descriptor record has the following format:

Word	
1	coded descriptor
2	coded descriptor. If word 1 = word 2, this is a primary descriptor. If word 2 is zero, this is a trouble term. Word 1 may never equal zero.
3	date of first appearance in any document profile or zero.
4	date of first appearance in any user profile or zero.
5	number of documents having this descriptor or zero.
6	number of users having this descriptor or zero.
7-26	alphanumeric descriptor, not over 120 characters or blanks.
27-30	not used. All zeros.

2. Document Profiles

This file is provided via other NASA operations, from which its format can be obtained. It is sorted on document number.

3. Coded User Profiles

The coded user profile tape consists of three files. Files 1 and 3 are labels. File 2 is sorted on user number and coded descriptor.

a. File 1 One record

Char.	
1-65	tape identification
66	tape number (1, 2, ...)
67-72	date tape was created

b. File 2 One record per user $\leq$ 5000 words

Word	
1-24	user header, card image
25	date of entry to system
26	date of last change to record
27	Bit
	1-12 number of single-word coded descriptors in sorted order, N
	13-24 number of added on latest run, M, not sorted
	25-36 number of deleted single-word descriptors, L
28-30	not used
31	reserved to indicate presence of 'not' descriptors

Following are M + N words of single-word coded descriptors:

Bit	
S-2	Octal 0, 'may' descriptor Octal 1, 'must' descriptor Octal 2, 'must' author Octal 3, 'must' contract number Octal 4, 'not' subject field Octal 5, 'not' author Octal 6, 'not' contract number Octal 7, 'not' descriptor
3-32	These thirty bits contain the coded value of the descriptor word or phrase.
33-35	For single-word descriptors, an octal 0 indicates deleted word, 1 that descriptor is used only as a single word, 2 that descriptor is used only in a phrase, 3 that descriptor is used both as a single word and in a phrase, 4 that descriptor is a special descriptor.

The remainder of the record consists of the coded values of the phrase descriptors and pointers to corresponding word locations in the single-word area. A given phrase would consist of one word of code, followed by as many words as are needed to contain the pointers, three to a word.

c. File 3 One record

Word	
1	Number of next user tape (0 if zero is last or only tape, otherwise 1, 2, ..)
2-14	Not used

4. Historical User Profiles (Optional)

The historical user profiles consist of three files. Files 1 and 3 are

labels. File 2 is sorted on user number.

a. File 1 One record

Char.	
1-65	Tape identification
66	Tape number (1, 2, ...)
67-72	Data tape was created

b. File 2 One record per user $\leq$ 1000 words

Record Format:

Word	
1-24	User header, card image
25	Date of entry to system
26	Date of deletion from system (zero if active)

Following is one logical record per descriptor in this format:

Word	
1	Coded value of descriptor
2	Date descriptor was added to profile
3	Date of deletion of descriptor (zero if active)
4	Length of this logical record, N
5 thru N-3	BCD descriptor, as many words as needed

c. File 3 One record

Word	
1	Number of next historical profiles tape (zero if last or only tape; otherwise 2, 3, ...)
2-14	Not used

5(a). Control Cards

Card 1	
Cols. 1-6	Label of input vocabulary. Slashes in 1-6 if no such input tape
Card 2	
Cols. 1-72	Label of output vocabulary control guide (used only if no input vocabulary file.)
Card 3	
Cols. 1-72	Label of output coded user profiles
Card 4	
Cols. 1-72	Label of input coded user profiles. Slashes in 1-6 if no such input tape.
Card 5	
Cols. 1-72	Label of input and output historical user profiles. Card 5 is blank if no historical profiles.
Card 6	Input tape unit assignment. First four tapes on one channel, next four on second channel.

A blank or zero assignment is not permitted.

Cols. 1-2	Input document profiles and input coded user profiles, reel 1
7-8	Output document profiles
13-14	Output vocabulary and output coded user profiles, reel 2
19-20	Input historical user profiles
25-26	Output historical user profiles
31-32	Input vocabulary and output coded user profiles, reel 1
37-38	System input tape
43-44	Input coded user profiles, reel 2
49-50	System output tape
Card 7	Tape numbers for which mounting will be requested on line. A blank indicates that no tape exists for this file. Tapes not mentioned are mounted at start of run.
Cols. 8-12	Input vocabulary
14-18	Input coded user profiles, reel 1
20-24	Input coded user profiles, reel 2
26-30	Input historical user profiles, reel 1
32-36	Input historical user profiles, reel 2
38-42	Output coded user profiles, reel 1
44-48	Output coded user profiles, reel 2
Card 8	Program parameters
Cols. 1-6	Date; numeric month, day, year
17-18	Phase number at which restart is to begin
22-24	Minimum percentage value of accepted descriptors for retention of user profile
28-30	Minimum percentage value of accepted descriptors for retention of document profile

If job is restarted, add to card 8 the following:

Col. 14	1 if document profile input, blank if none
16	1 if coded user profile input or vocabulary EQUATE changes, blank if neither

Following the eight control cards are cards for vocabulary and user changes.

Following those, if any, is one of two cards:

Cols. 1-6	NODOCT if there is no input document profile tape.
1-7	DOCTAP1 if there is an input document profile tape. DOCTAP2 if there are two input document profile tapes.

5(b). Vocabulary Changes

Card 1	
Cols. 1-4	DICT

Descriptors may be added to or deleted from the vocabulary, classified as trouble terms, or equated as synonyms. The trouble terms and equate changes will also add descriptors to the vocabulary if the descriptors involved are not already there.

a. Add Changes

Card 1		
Cols.	1-60	Alphameric descriptor
	71	A
	72	Blank if no second card, C if second card follows
Card 2		
Cols.	1-60	Alphameric descriptor continued
	71	A
	72	Blank

b. Delete Changes

Same as add changes, except D in col. 71.

c. Trouble Term Change

Same as add changes, except * in col. 71.

d. Equate Change (will cause updating of user profiles)

Card	1	Secondary
Cols.	1-60	Alphameric descriptor, secondary
	71	E
	72	Blank if no second card, C if second card follows
Card 2		Secondary
Cols.	1-60	Alphameric descriptor, secondary continued
	71	E
	72	Blank
Card 3		Primary
Cols.	1-60	Alphameric descriptor, primary
	71	T
	72	Blank if no second card, C if second card follows
Card 4		Primary
Cols.	1-60	Alphameric descriptor, primary, continued
	71	T
	72	Blank
Last Card		
Cols.	1-6	ENDICT

5(c). New user profiles and user profile changes.

Card 1
Cols. 1-4 USER

a. New Users - Header

Card 1

Cols. 1- 6	Profile number
8, 10	User's initials
12-30	User's surname
36-38	Location
44-48	Employee number (optional)
52	Corporate security clearance
53	Federal security clearance
70	U
71	H
72	Blank if one header card, otherwise C

Card 2

Cols. 1- 6	Same as first card
7-66	Address for outside mailing (left adjusted)
70-71	Same as first card
72	Blank

b. New Users - Descriptors

Card 1

Cols. 1-60	Alphameric descriptor
61-66	Profile number
70	Descriptor usage code
71	Blank
72	C if second card follows, blank if no second card

Card 2

Cols. 1-60	Alphameric descriptor, continued
61-66	Profile number
70	Descriptor usage code
71	Blank
72	Blank

Where descriptor usage code is:

M	'must' ordinary descriptor
N	'not' ordinary descriptor
C	'must' contract-number
D	'not' contract number
P	'must' author
Q	'not' author
S	'not' subject field

c. Add Changes - Header
Same as new users

d. Add Changes - Descriptors
Same as new users, except A in col. 71.

e. Delete Changes - Descriptors
Same as new users, except D in col. 71.

f. Delete entire user profile

Cols.	1-60	Blank
	61-66	Profile number
	67-70	Blank
	71	D
	72	U

Last Card

Cols. 1-6	ENDUSR

If historical user profiles are kept, user reports may be called for between the USER and ENDUSR cards.

g. User Reports

Cols. 1- 6	Lower profile limit of report, profile number
8-13	Upper profile limit of report, profile number, or blank if only one profile (col. 1-6) to be reported
71	R
72	1 if only active descriptors are to be reported 2 if all descriptors are to be reported

2.2 Output

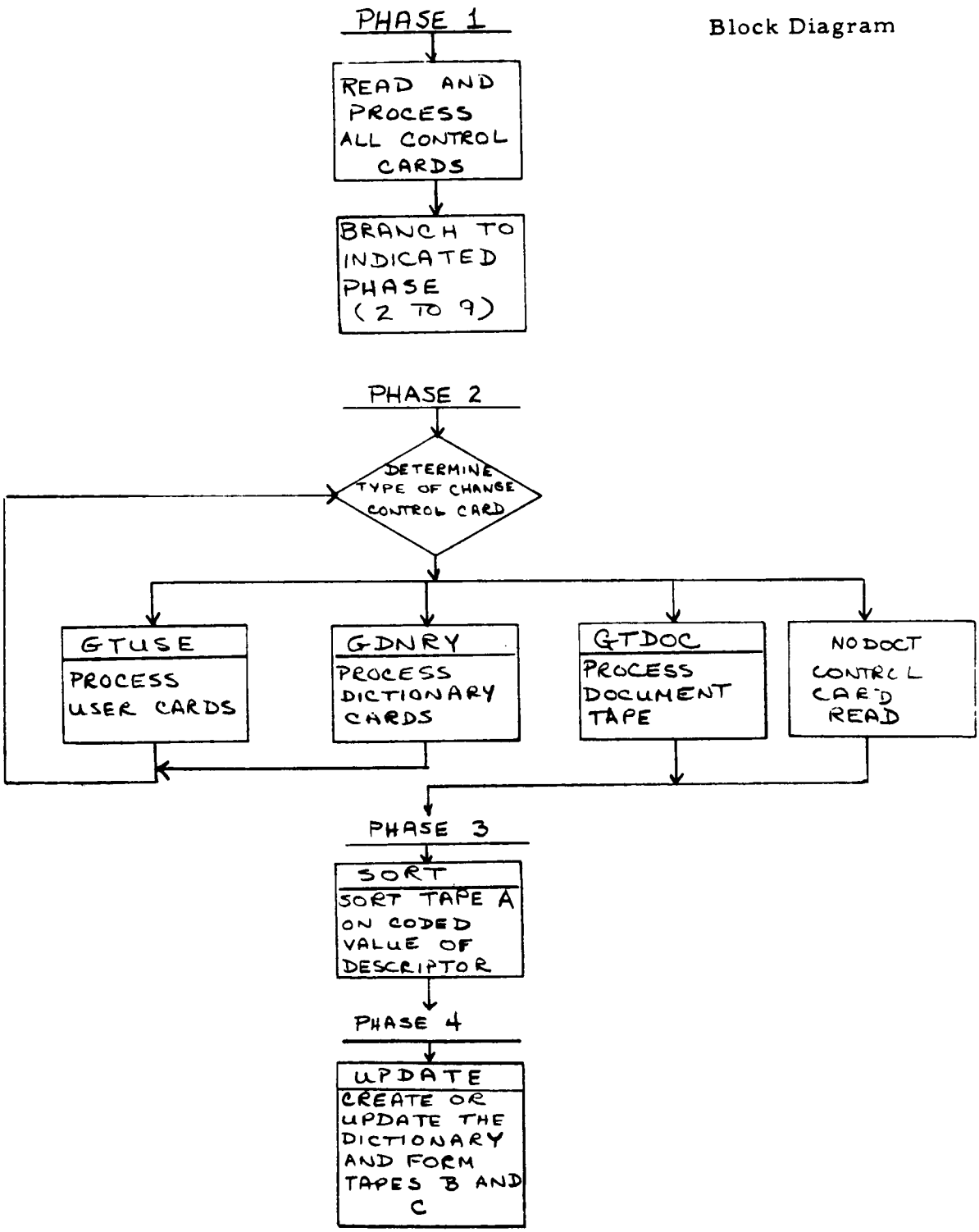
1. Vocabulary control guide, updated - same format as input.
2. Coded document profiles - same format as coded user profiles.
3. Coded user profiles, updated - same format as input.
4. Historical user profiles, updated (optional) - same format as input.
5. Operational comments - see example following program listing.

MAIN PROGRAM

This program is divided into nine phases. In phase 1, all control cards are read and processed. In phase 2, user cards are processed by GTUSE, vocabulary cards by GDNRY, and the document tape by GTDOC. The output tape resulting from this phase is called tape A. During phase 3, tape A is sorted on the coded value of the descriptors. Phase 4 uses the UPDATE subroutine to create or update the vocabulary tape while at the same time forming document tape (tape B) and/or a user tape (tape C). Phase 5 sorts the document tape according to profile and serial number, and phase 6 sorts the user tape according to profile and serial number. During phase 7, a document profile tape is created. During phase 8, a user profile tape is created or updated, and also an historical user profile tape may be created or updated. Phase 9 summarizes program activity, then exits. The subroutine of the program follow, in alphabetical order.

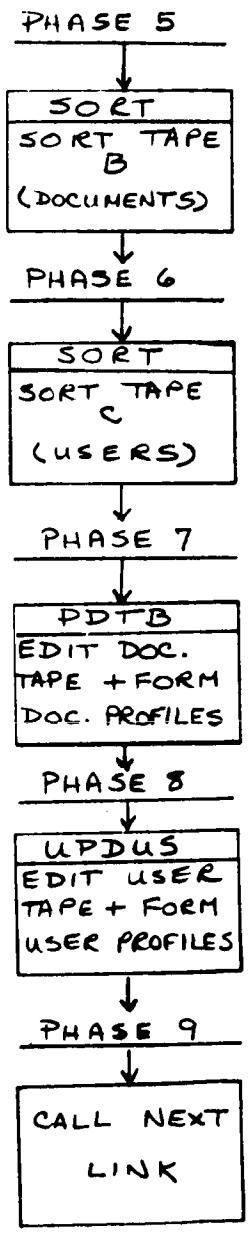
MAIN 1/2

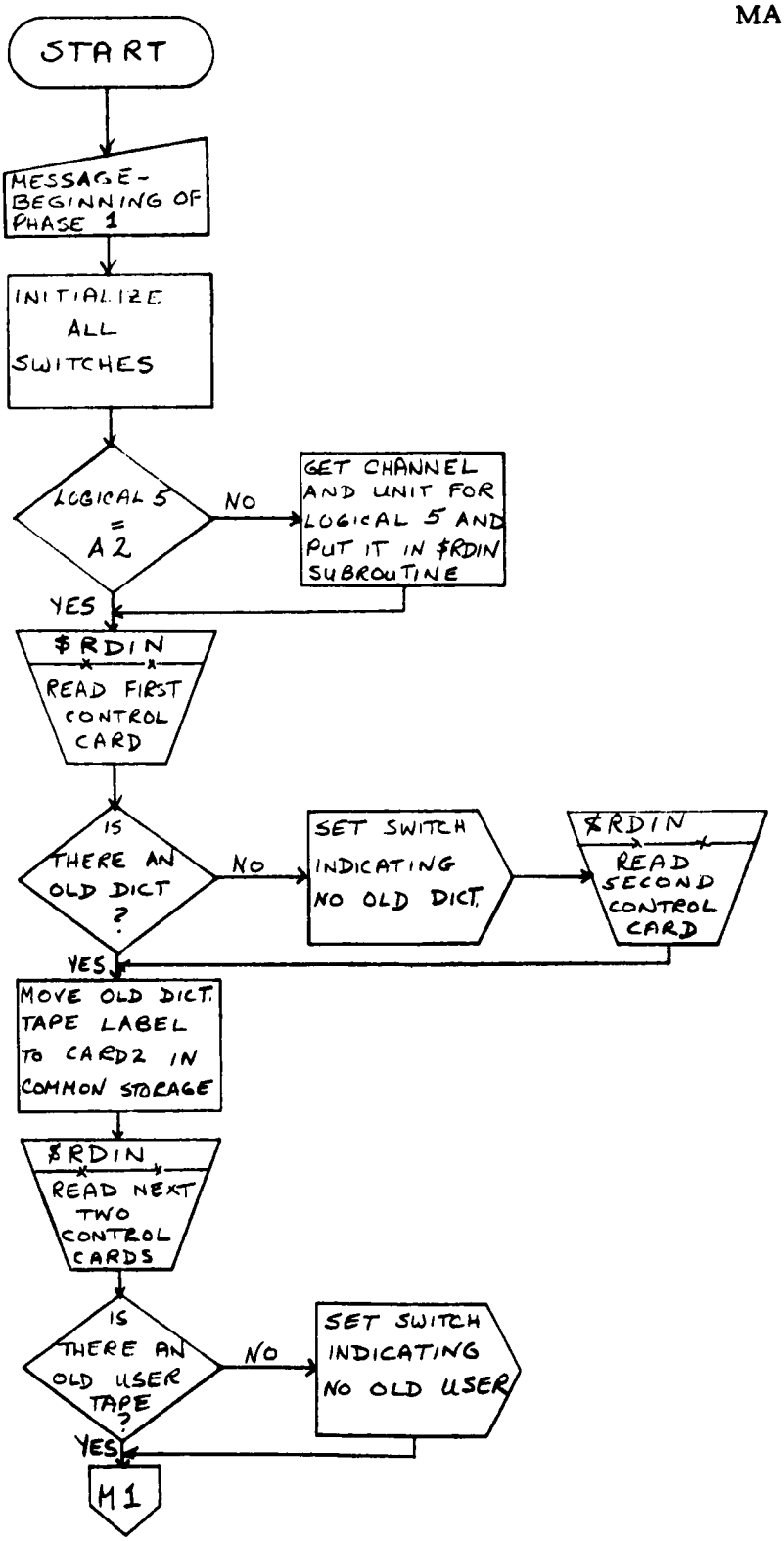
Block Diagram

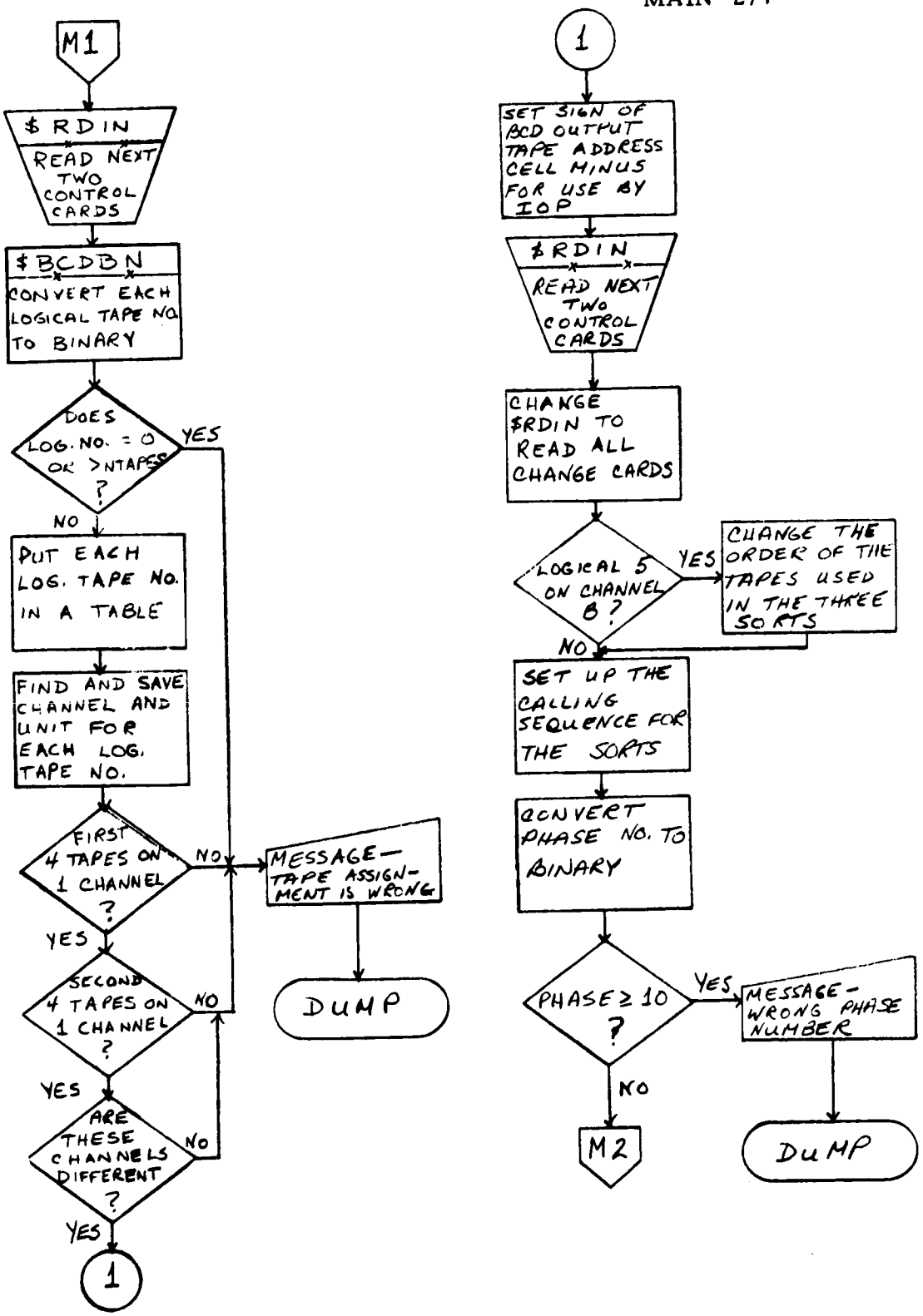


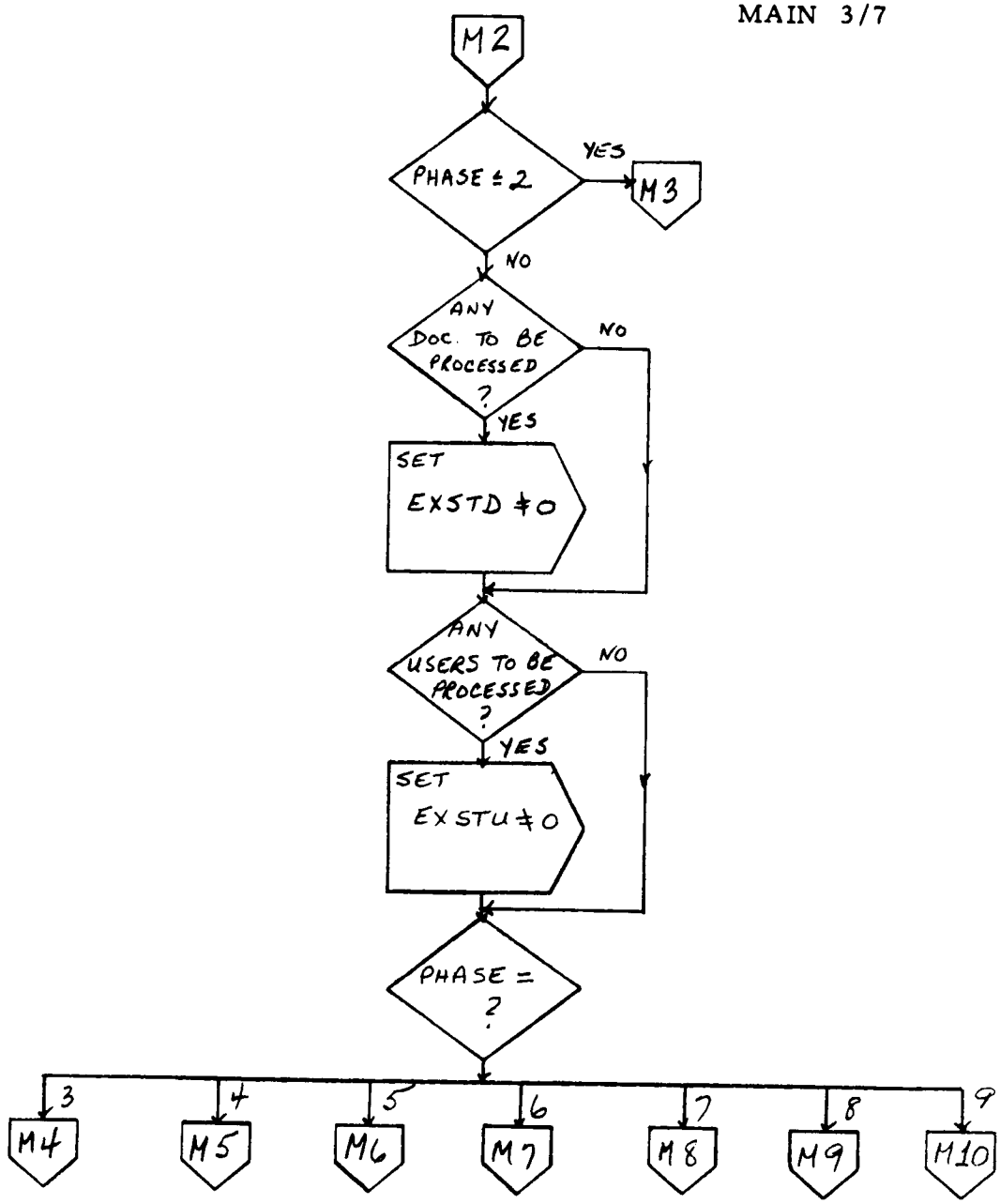
MAIN 2/2

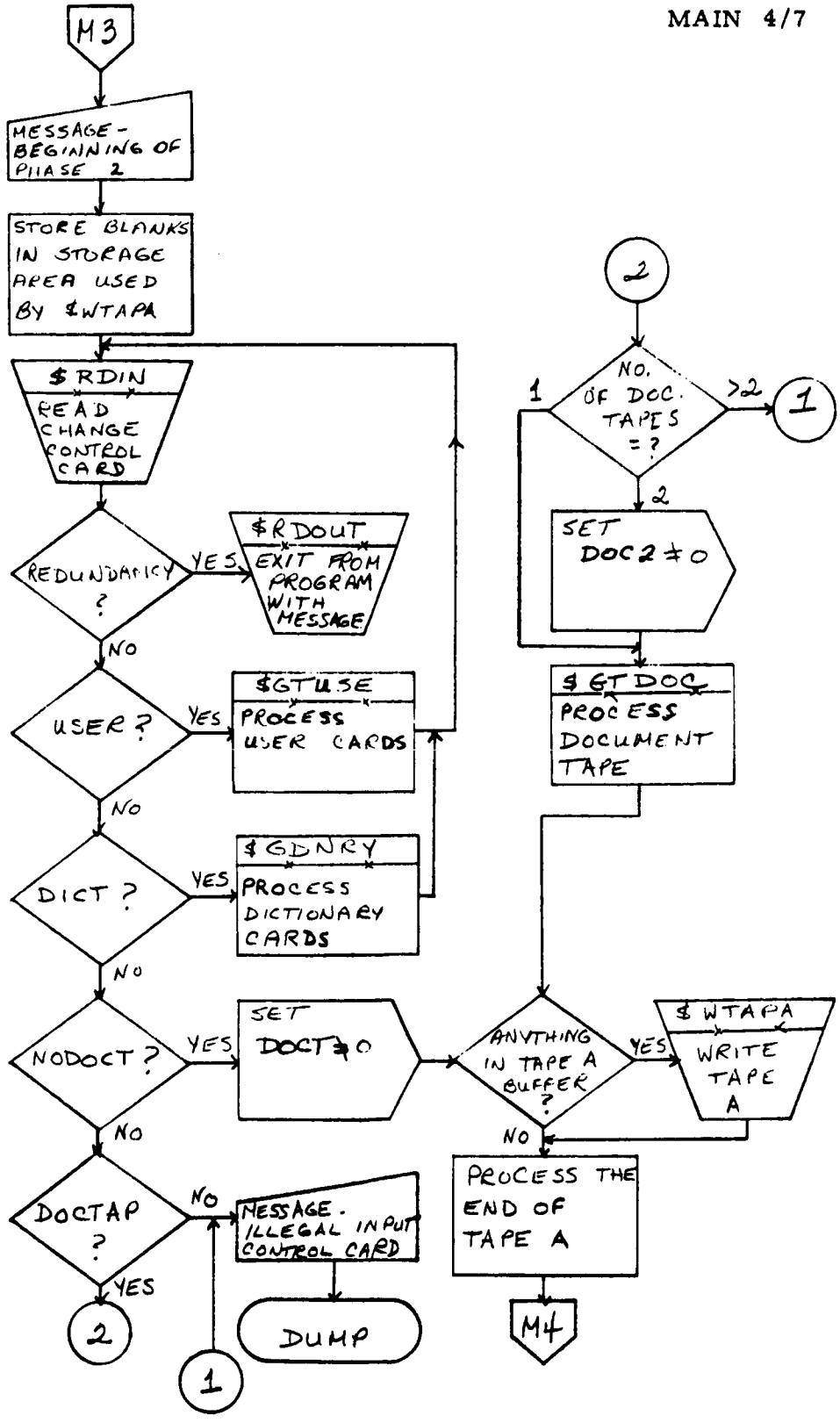
Block Diagram

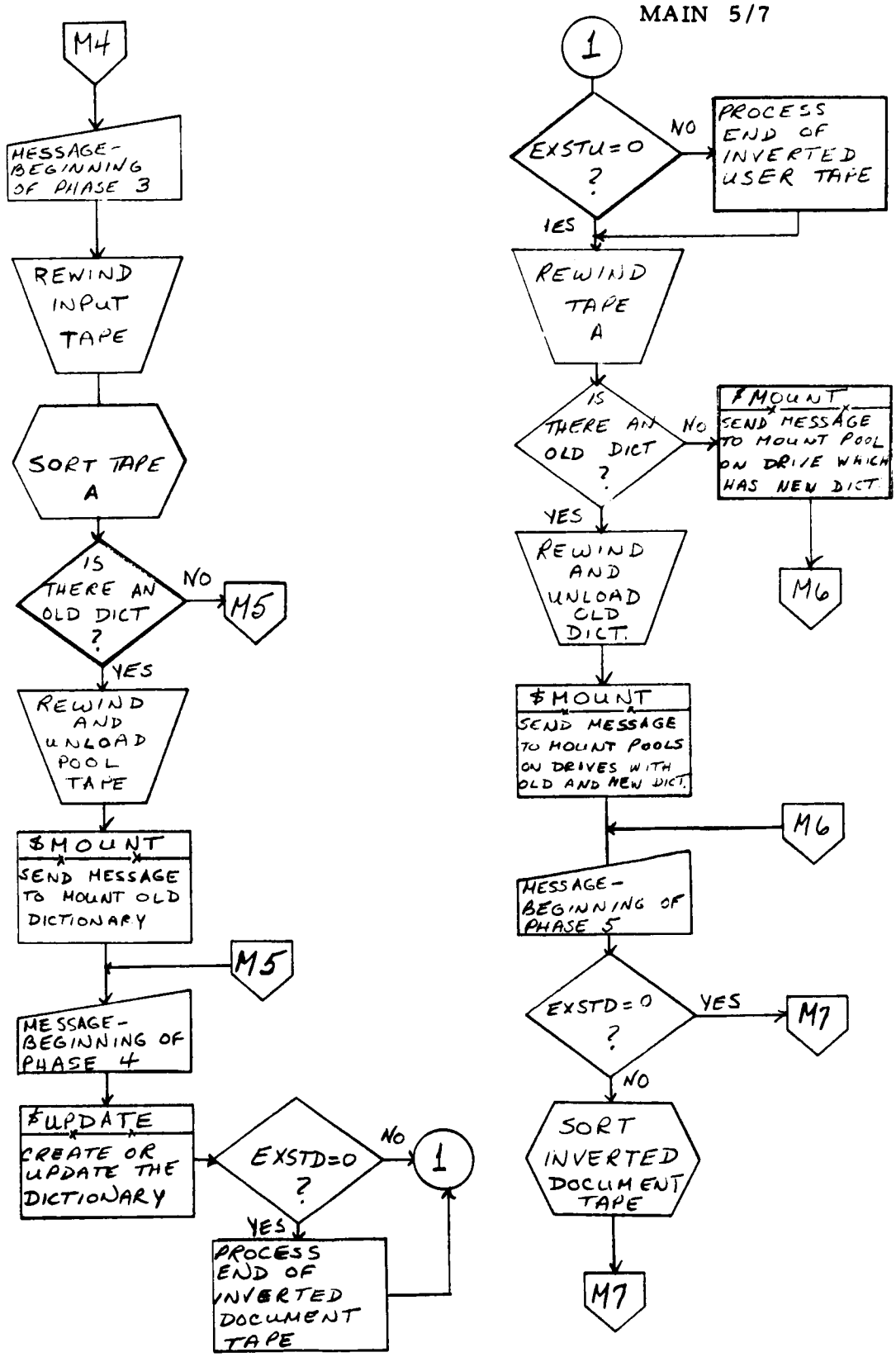


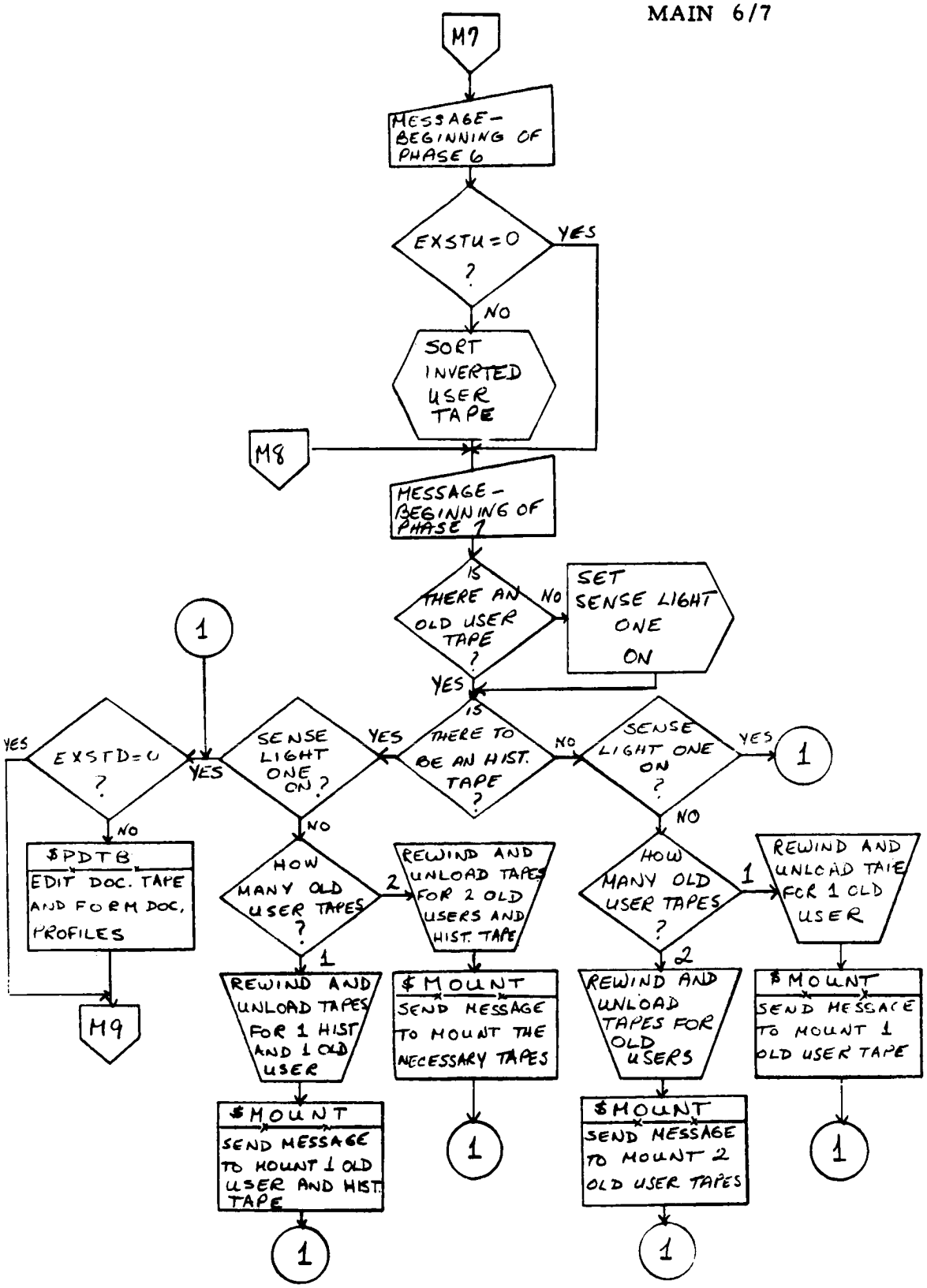

START
MESSAGE - BEGINNING OF PHASE 1
INITIALIZE ALL SWITCHES
LOGICAL 5 = A2
NO
GET CHANNEL AND UNIT FOR LOGICAL 5 AND PUT IT IN $RDIN SUBROUTINE
YES
$RDIN
READ FIRST CONTROL CARD
IS THERE AN OLD DICT ?
NO
SET SWITCH INDICATING NO OLD DICT.
$RDIN
READ SECOND CONTROL CARD
YES
MOVE OLD DICT. TAPE LABEL TO CARD2 IN COMMON STORAGE
$RDIN
READ NEXT TWO CONTROL CARDS
IS THERE AN OLD USER TAPE ?
NO
SET SWITCH INDICATING NO OLD USER
YES
M1

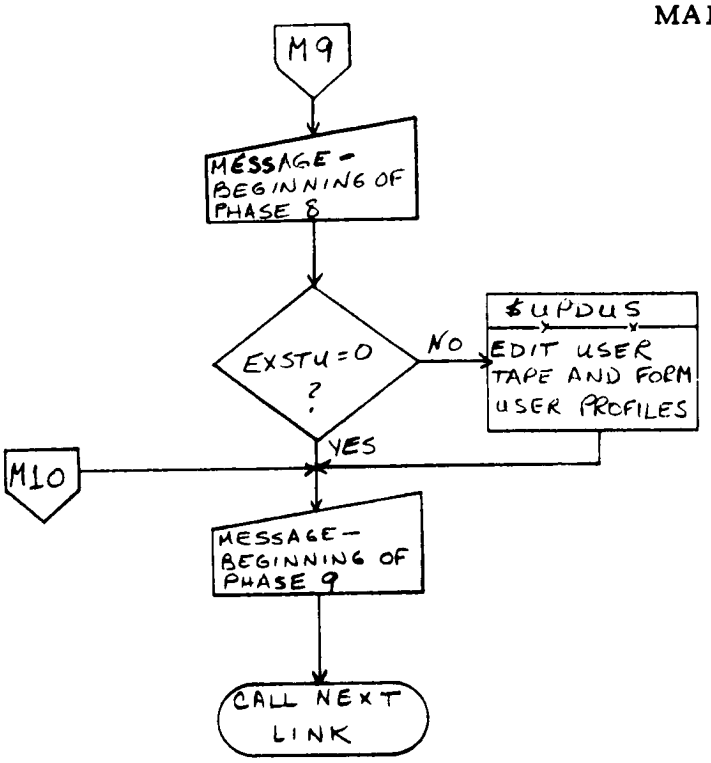

M1
$RDIN
READ NEXT TWO CONTROL CARDS
$BCDBN
CONVERT EACH LOGICAL TAPE NO. TO BINARY
DOES LOG. NO. = 0 OR >NTAPES ?
YES
NO
PUT EACH LOG. TAPE NO. IN A TABLE
FIND AND SAVE CHANNEL AND UNIT FOR EACH LOG. TAPE NO.
FIRST 4 TAPES ON 1 CHANNEL ?
NO
MESSAGE— TAPE ASSIGN-MENT IS WRONG
DUMP
YES
SECOND 4 TAPES ON 1 CHANNEL ?
NO
YES
ARE THESE CHANNELS DIFFERENT ?
NO
YES
1
1
SET SIGN OF BCD OUTPUT TAPE ADDRESS CELL MINUS FOR USE BY IOP
$RDIN
READ NEXT TWO CONTROL CARDS
CHANGE $RDIN TO READ ALL CHANGE CARDS
LOGICAL 5 ON CHANNEL B ?
YES
CHANGE THE ORDER OF THE TAPES USED IN THE THREE SORTS
NO
SET UP THE CALLING SEQUENCE FOR THE SORTS
CONVERT PHASE NO. TO BINARY
PHASE ≥ 10 ?
YES
MESSAGE— WRONG PHASE NUMBER
NO
M2
DUMP


M2
PHASE ≠ 2
YES
M3
NO
ANY DOC. TO BE PROCESSED ?
NO
YES
SET EXSTD ≠ 0
ANY USERS TO BE PROCESSED ?
NO
YES
SET EXSTU ≠ 0
PHASE = ?
3
4
5
6
7
8
9
M4
M5
M6
M7
M8
M9
M10


M3
MESSAGE - BEGINNING OF PHASE 2
STORE BLANKS IN STORAGE AREA USED BY $WTAPA
$RDIN
READ CHANGE CONTROL CARD
REDUNDANCY ?
YES
$RDOUT
EXIT FROM PROGRAM WITH MESSAGE
NO
USER ?
YES
$GTUSE
PROCESS USER CARDS
NO
DICT ?
YES
$GDNRY
PROCESS DICTIONARY CARDS
NO
NODOCT ?
YES
SET DOCT≠0
NO
DOCTAP ?
NO
MESSAGE - ILLEGAL INPUT CONTROL CARD
DUMP
YES
2
1
2
NO. OF DOC. TAPES = ?
1
>2
1
2
SET DOC2≠0
$GTDOC
PROCESS DOCUMENT TAPE
ANYTHING IN TAPE A BUFFER ?
YES
$WTAPA
WRITE TAPE A
NO
PROCESS THE END OF TAPE A
M4

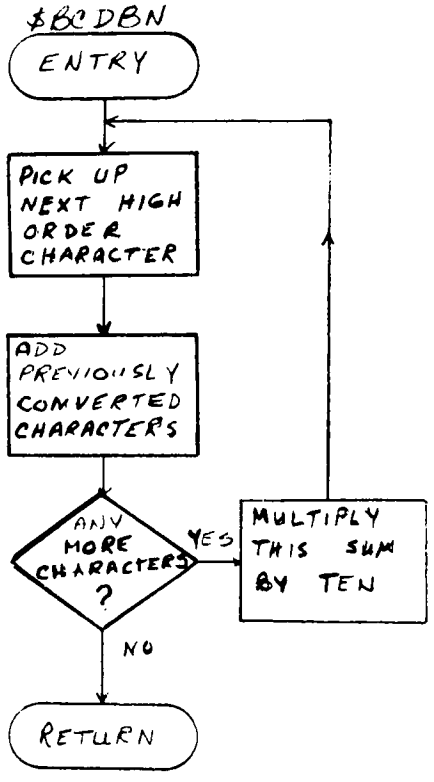

M4
MESSAGE-BEGINNING OF PHASE 3
REWIND INPUT TAPE
SORT TAPE A
IS THERE AN OLD DICT ?
NO
M5
YES
REWIND AND UNLOAD POOL TAPE
$MOUNT
SEND MESSAGE TO MOUNT OLD DICTIONARY
M5
MESSAGE-BEGINNING OF PHASE 4
$UPDATE
CREATE OR UPDATE THE DICTIONARY
EXSTD=0 ?
NO
1
YES
PROCESS END OF INVERTED DOCUMENT TAPE
1
EXSTU=0 ?
NO
PROCESS END OF INVERTED USER TAPE
YES
REWIND TAPE A
IS THERE AN OLD DICT ?
NO
$MOUNT
SEND MESSAGE TO MOUNT POOL ON DRIVE WHICH HAS NEW DICT.
M6
YES
REWIND AND UNLOAD OLD DICT.
$MOUNT
SEND MESSAGE TO MOUNT POOLS ON DRIVES WITH OLD AND NEW DICT.
M6
MESSAGE-BEGINNING OF PHASE 5
EXSTD=0 ?
YES
M7
NO
SORT INVERTED DOCUMENT TAPE
M7


M7
MESSAGE - BEGINNING OF PHASE 6
EXSTU=0 ?
YES
NO
SORT INVERTED USER TAPE
M8
MESSAGE - BEGINNING OF PHASE 7
IS THERE AN OLD USER TAPE ?
NO
SET SENSE LIGHT ONE ON
YES
1
YES
EXSTD=0 ?
YES
NO
$PDTB
EDIT DOC. TAPE AND FORM DOC. PROFILES
M9
SENSE LIGHT ONE ON ?
YES
NO
HOW MANY OLD USER TAPES ?
2
1
REWIND AND UNLOAD TAPES FOR 2 OLD USERS AND HIST. TAPE
REWIND AND UNLOAD TAPES FOR 1 HIST AND 1 OLD USER
$MOUNT
SEND MESSAGE TO MOUNT THE NECESSARY TAPES
$MOUNT
SEND MESSAGE TO MOUNT 1 OLD USER AND HIST TAPE
1
1
IS THERE TO BE AN HIST. TAPE ?
NO
SENSE LIGHT ONE ON ?
YES
1
NO
HOW MANY OLD USER TAPES ?
1
2
REWIND AND UNLOAD TAPE FOR 1 OLD USER
REWIND AND UNLOAD TAPES FOR OLD USERS
$MOUNT
SEND MESSAGE TO MOUNT 1 OLD USER TAPE
$MOUNT
SEND MESSAGE TO MOUNT 2 OLD USER TAPES
1
1


M9
MESSAGE - BEGINNING OF PHASE 8
EXSTU=0 ?
NO
$UPDUS
EDIT USER TAPE AND FORM USER PROFILES
YES
M10
MESSAGE - BEGINNING OF PHASE 9
CALL NEXT LINK

Subroutine BCDBN

This subroutine converts a BCD integer into a binary number.


$BCDBN
ENTRY
PICK UP NEXT HIGH ORDER CHARACTER
ADD PREVIOUSLY CONVERTED CHARACTERS
ANY MORE CHARACTERS ?
YES
MULTIPLY THIS SUM BY TEN
NO
RETURN

Subroutine CODER

The calling sequence of CODER is

```
        TSX     $CODER,4
        TSX     TYPE,0
        TSX     BUFIN,0
        TSX     BUFOUT,0
        TSX     K,0
```

where

TYPE indicates source of input
- 0 vocabulary change
- 1 document descriptor
- 3 new user descriptor
- 4 user change descriptor.

BUFIN is location of start of input buffer.

BUFOUT is location of start of output buffer. BUFIN and BUFOUT arrays are in forward, FAP order.

K is the number of sort items placed in BUFOUT by CODER. If the input descriptor consists of n alphameric words, then K = n if n = 1, or K = n + 1 if n > 1.

CODER expects 72 characters per input card image, the alphameric cols. 1-60. Profile header cards are not input to CODER. Each descriptor input to CODER will generate K items of the format:

Word

1 coded value of input descriptor

2 alphameric profile number (zero if TYPE = 0)

3 miscellaneous:

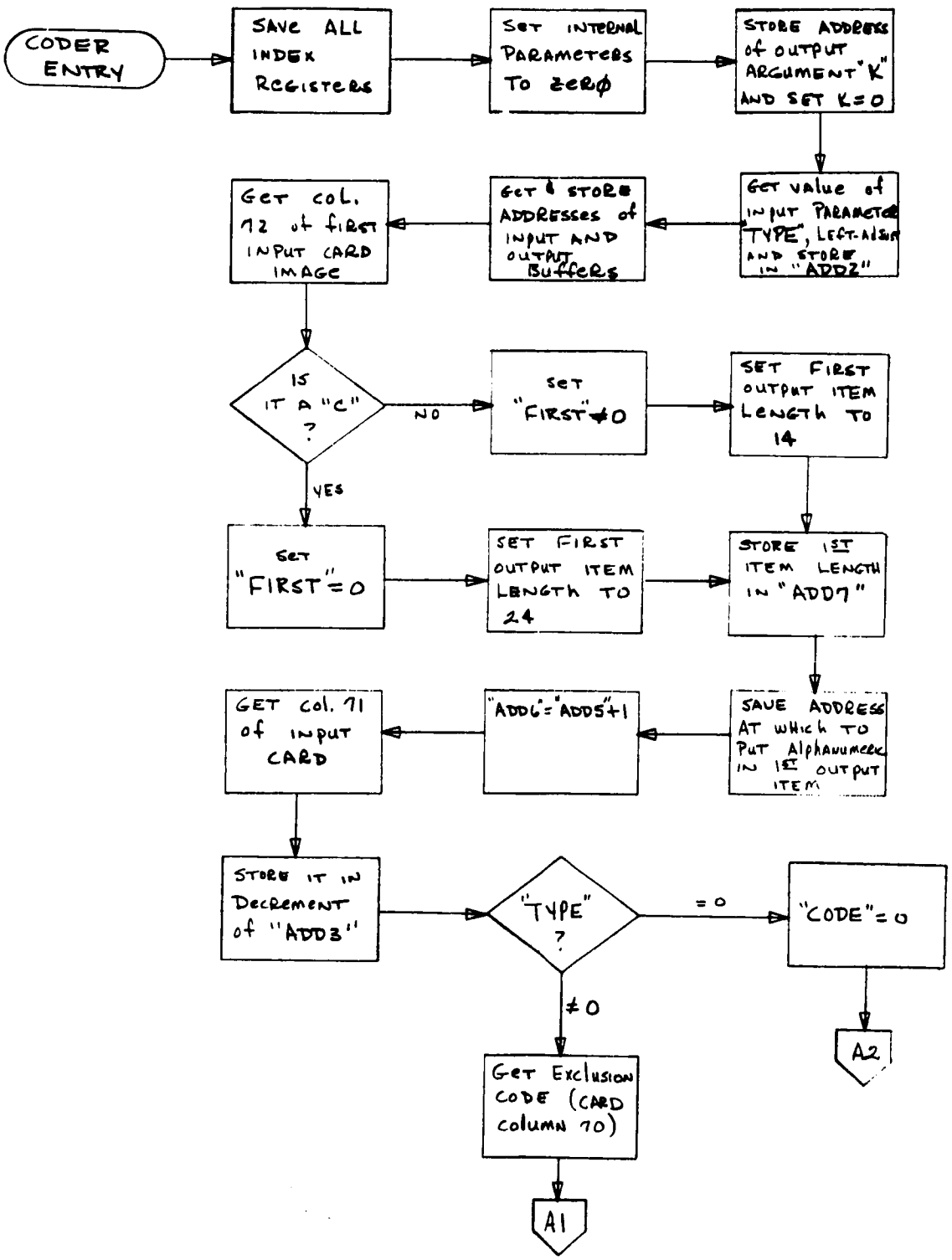
Bit	
S- 2	zero
3- 5	same as TYPE
6-11	same as col. 71 of input card
12-13	0 do not add to vocabulary
	1 add to vocabulary
14-15	0 do not add to profile if code not on vocabulary
	1 add code to profile
	3 add code to profile descriptor list
16	reserved for use by subroutine UPDATE
17	determined by the subroutine that calls CODER.
	1 if code for word within phrase
	0 if single word descriptor or coded phrase
18-35	zero.

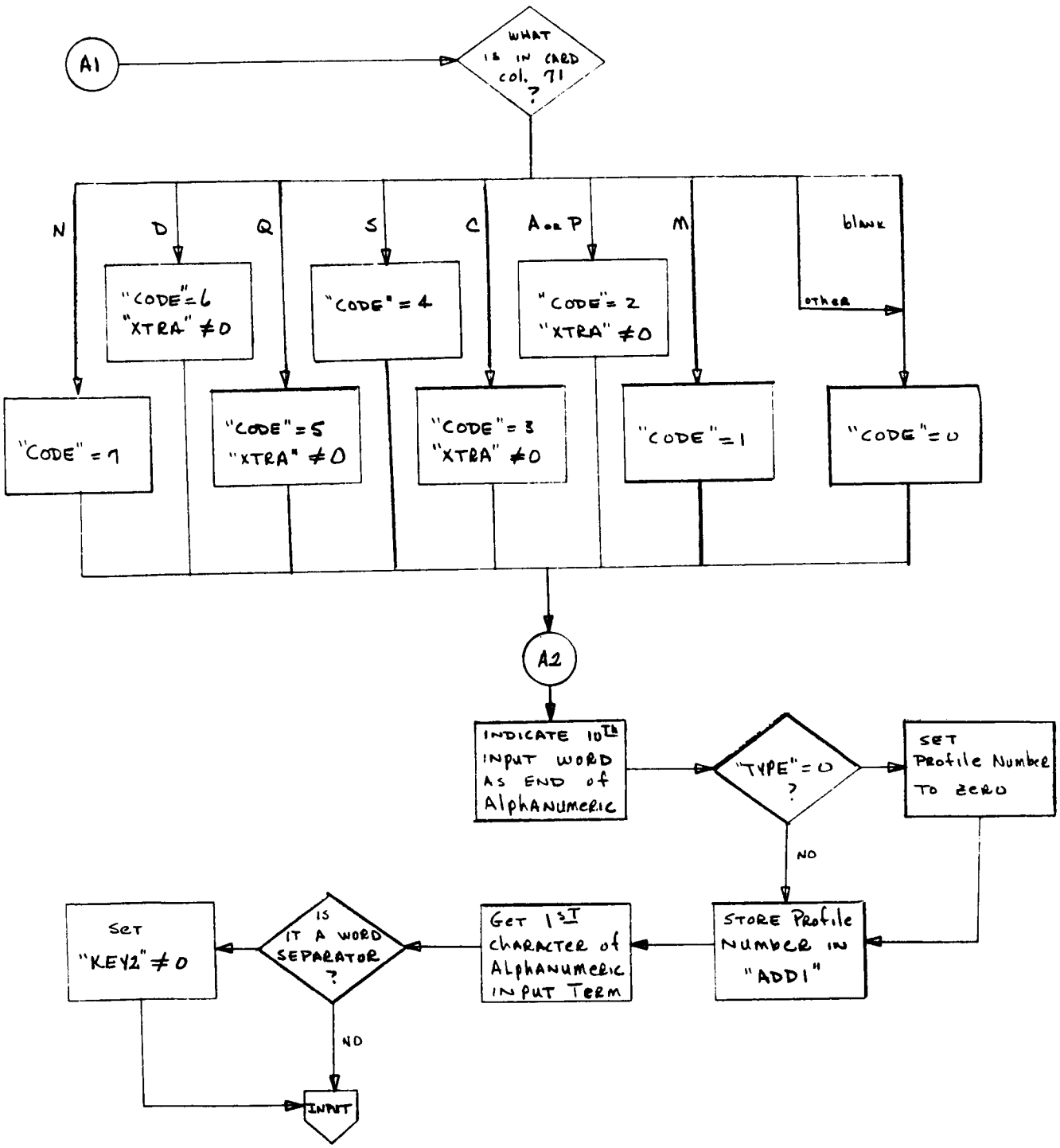
4 contains L, length of item in computer words.

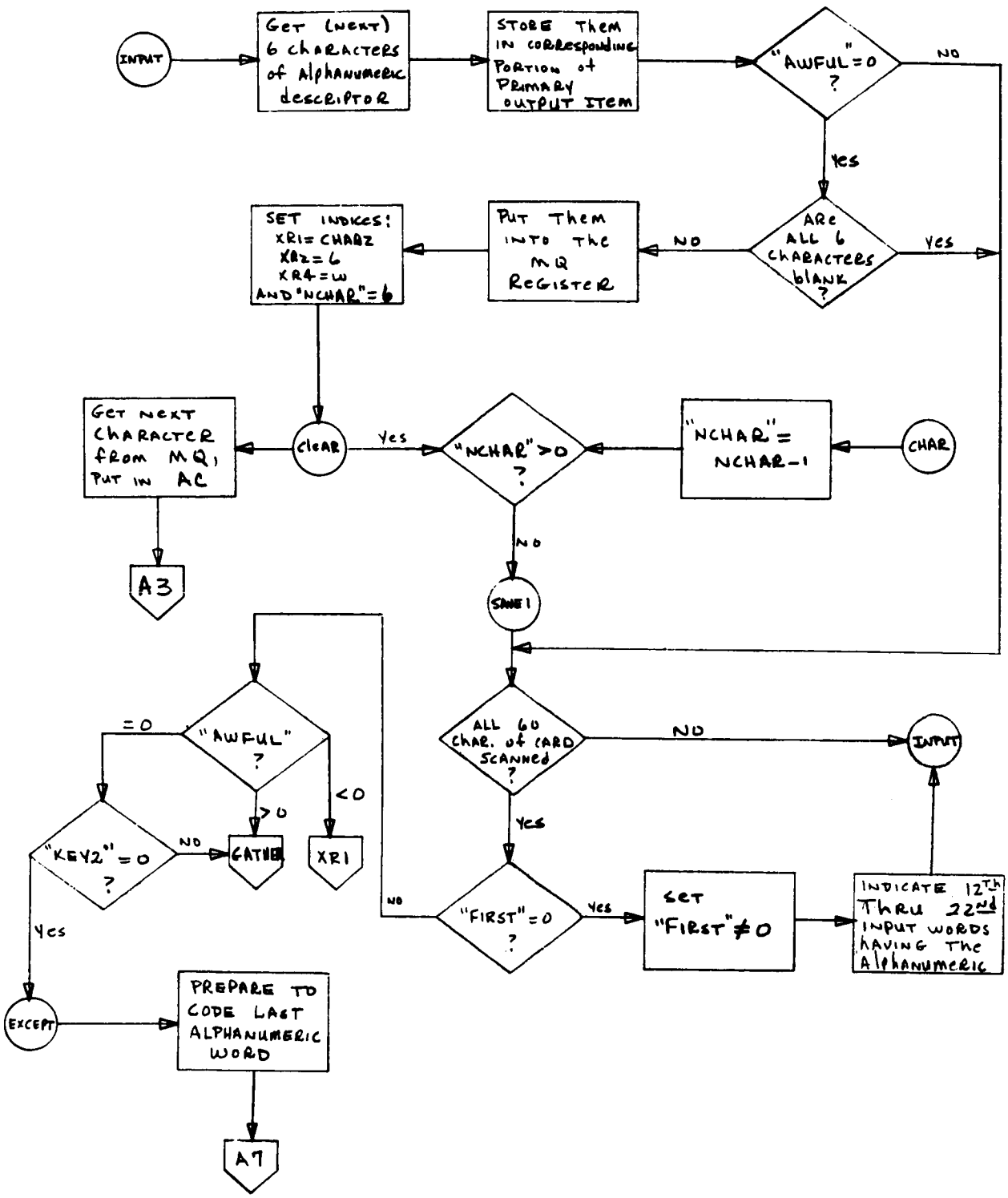
5-L alphameric version of the coded input.

If K>1, the second and following output items are four (L = 4) words in length and have no alphameric version. The first item contains the alphameric version and is 14 or 24 words in length (L = 14 or 24). If a descriptor is illegal, i.e., contains an illegal character, then K = 1, word 1 of the item is all zero, and word 3 is set:

Bit	
12-14	zero
18-35	77777)8


CODER ENTRY
SAVE ALL INDEX REGISTERS
SET INTERNAL PARAMETERS TO zero
STORE ADDRESS of OUTPUT ARGUMENT "K" AND SET K=0
GET value of INPUT PARAMETER "TYPE", LEFT-ADJUST AND STORE IN "ADD2"
GET & STORE ADDRESSES of INPUT AND OUTPUT Buffers
GET col. 72 of FIRST INPUT CARD IMAGE
IS IT A "C" ?
NO
SET "FIRST"≠0
SET FIRST OUTPUT ITEM LENGTH TO 14
YES
SET "FIRST"=0
SET FIRST OUTPUT ITEM LENGTH TO 24
STORE 1ST ITEM LENGTH IN "ADD7"
SAVE ADDRESS AT WHICH TO PUT ALPHANUMERIC IN 1ST OUTPUT ITEM
"ADD6"="ADD5"+1
GET col. 71 of INPUT CARD
STORE IT IN DECREMENT of "ADD3"
"TYPE" ?
=0
"CODE"=0
A2
≠0
GET EXCLUSION CODE (CARD COLUMN 70)
A1


A1
WHAT IS IN CARD col. 71 ?
N
D
Q
S
C
A or P
M
blank
other
"CODE" = 6
"XTRA" ≠ 0
"CODE" = 4
"CODE" = 2
"XTRA" ≠ 0
"CODE" = 7
"CODE" = 5
"XTRA" ≠ 0
"CODE" = 3
"XTRA" ≠ 0
"CODE" = 1
"CODE" = 0
A2
INDICATE 10TH INPUT WORD AS END of AlphANumeric
"TYPE" = 0 ?
SET Profile Number TO ZERO
NO
STORE Profile Number IN "ADD1"
GET 1ST CHARACTER of AlphANumeric INPUT Term
IS IT A WORD SEPARATOR ?
SET "KEY2" ≠ 0
NO
INPT

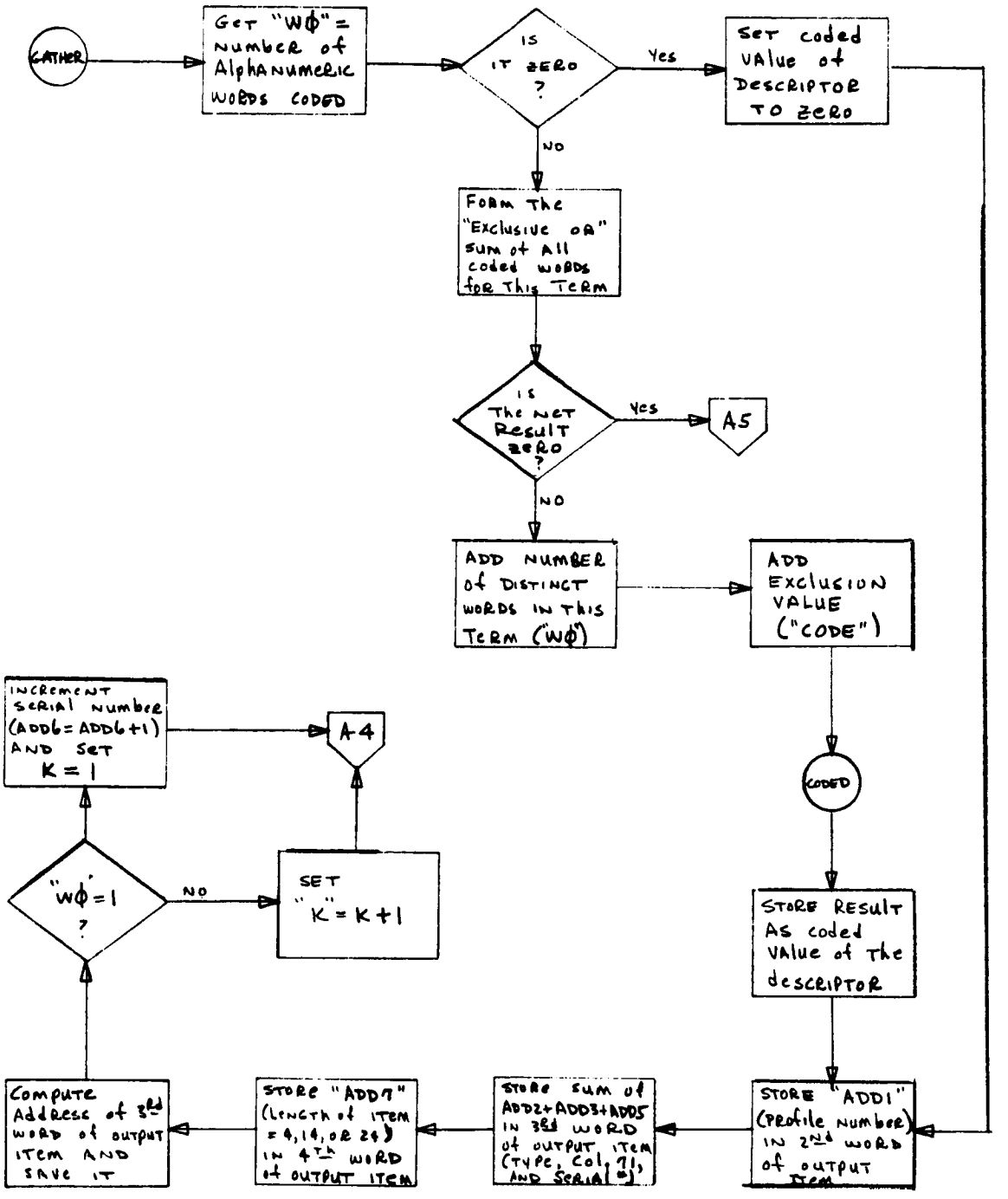

INPUT
Get (next) 6 characters of Alphanumeric descriptor
STORE them in corresponding Portion of PRIMARY OUTPUT ITEM
"AWFUL" = 0 ?
NO
Yes
ARe ALL 6 CHARACTERS blANK ?
Yes
NO
Put them into the MQ Register
SET INDICES: XR1 = CHAR2 XR2 = 6 XR4 = ω AND "NCHAR" = 6
Get next CHARACTER from MQ, PUT IN AC
CLEAR
Yes
"NCHAR" > 0 ?
"NCHAR" = NCHAR - 1
CHAR
A3
NO
SAME1
ALL 60 CHAR. of CARD SCANNED ?
NO
INPUT
Yes
"AWFUL" ?
= 0
> 0
< 0
"KEY2" = 0 ?
NO
GATHER
XR1
Yes
"FIRST" = 0 ?
NO
Yes
SET "FIRST" ≠ 0
INDICATE 12TH THRU 22nd INPUT WORDS HAVING The Alphanumeric
EXCEPT
PREPARE TO CODE LAST ALPHANUMERIC WORD
A7

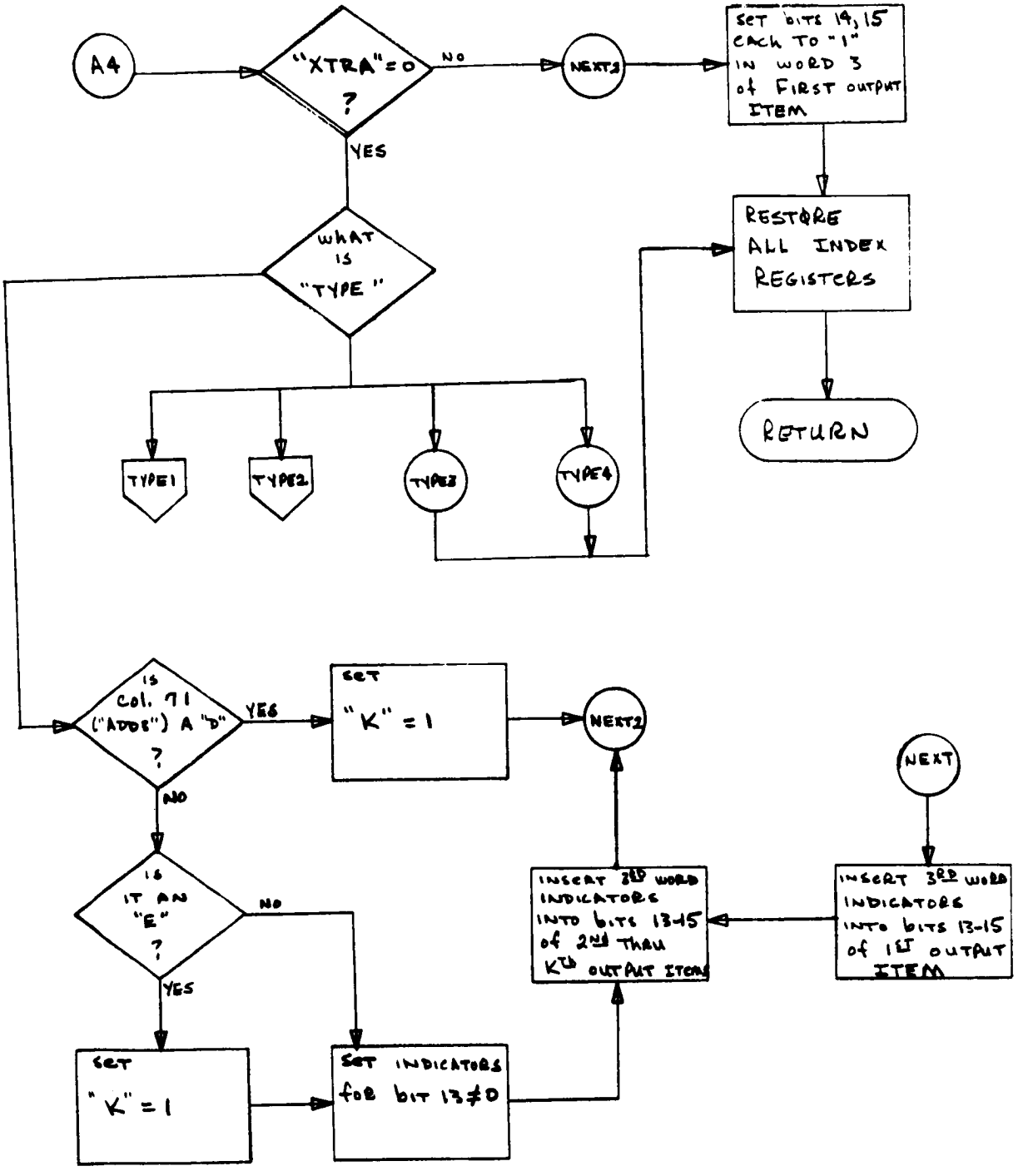
(A3) Table of Transfers Dependent on Current Character in Accumulator (AC)

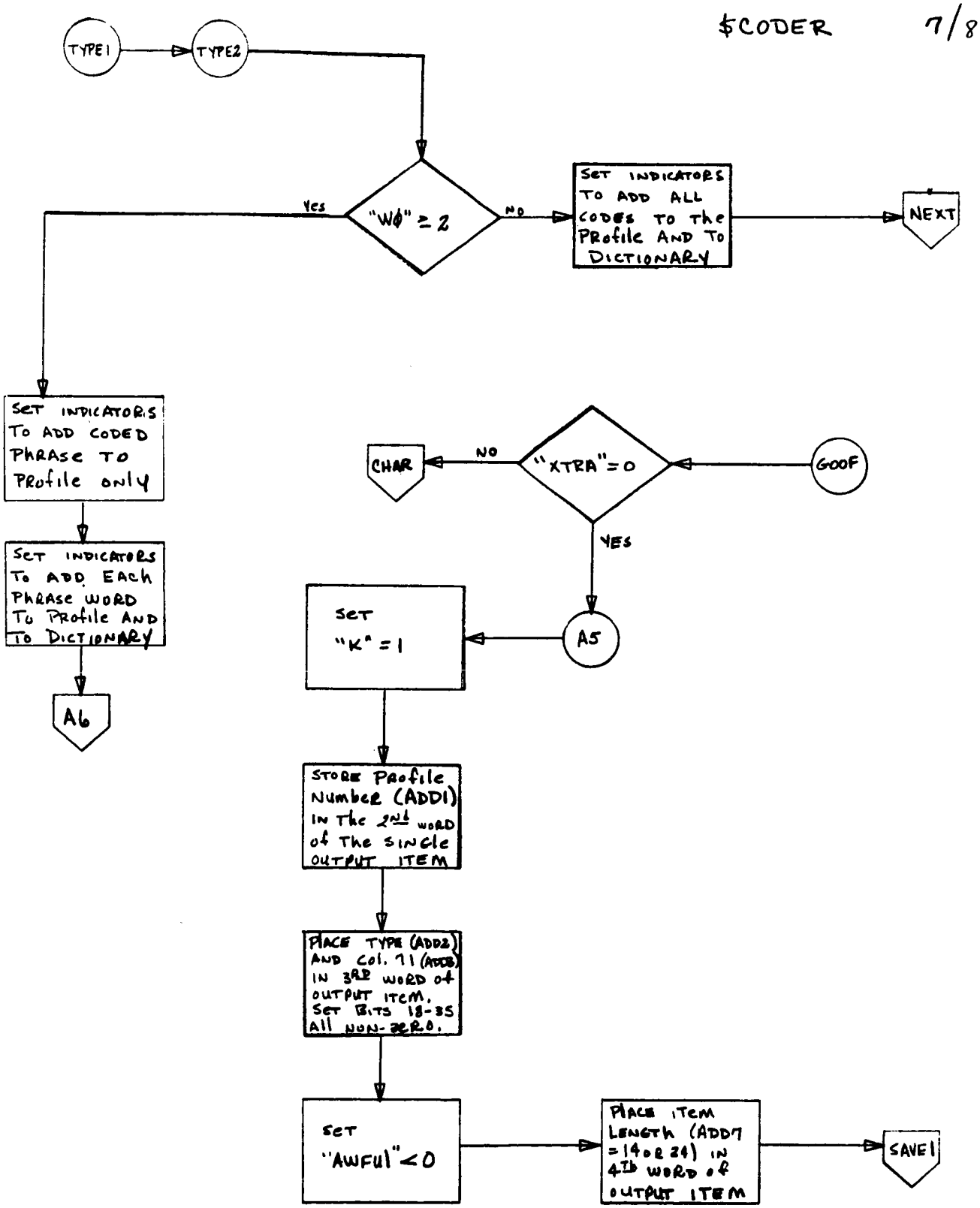
If character is:		Go to
00	0	SAME
01	1	
02	2	
03	3	
04	4	
05	5	
06	6	
07	7	
10	8	
11	9	
12		GOOF
13	=	"
14	-	CHAR
15		GOOF
16		
17		
20		SAME
21	A	
22	B	
23	C	
24	D	
25	E	
26	F	
27	G	
30	H	
31	I	
32		GOOF
33	.	CHAR
34	)	GOOF
35		
36		
37		

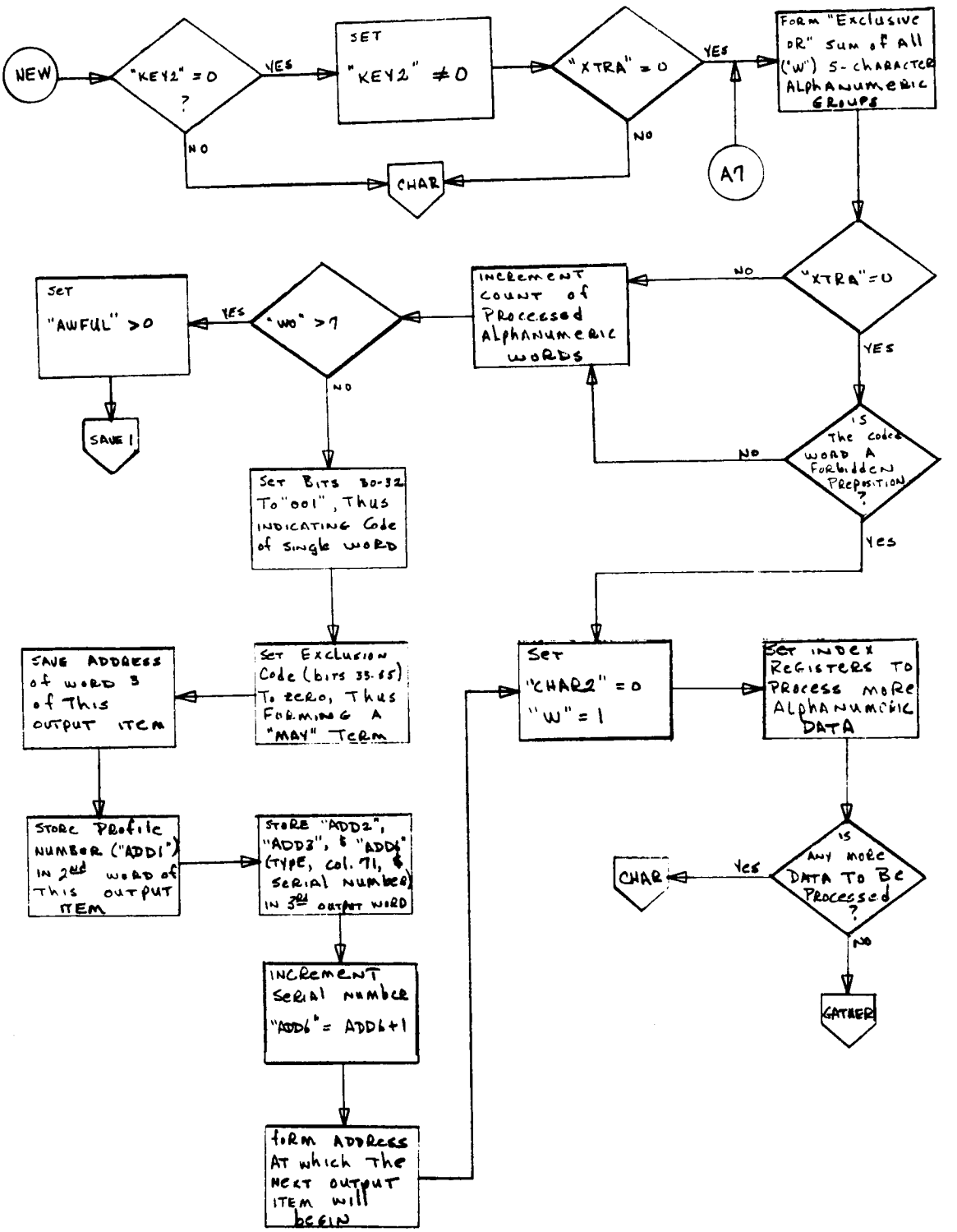
If character is:		Go to
40	-	CHAR
41	J	SAME
42	K	
43	L	
44	M	
45	N	
46	O	
47	P	
50	Q	
51	R	
52		GOOF
53	$	"
54	*	NEW
55		GOOF
56		
57		
60	Blank	NEW
61	/	"
62	S	SAME
63	T	
64	U	
65	V	
66	W	
67	X	
70	Y	
71	Z	
72		GOOF
73	,	NEW
74	(	GOOF
75		
76		
77		

Go to "SAME" if legitimate character of a word.
Go to "GOOF" if character is illegal.
Go to "CHAR" if character is to be ignored.
Go to "NEW" if character is a word separator.


GATHER
Get "WØ" = number of AlphaNumeric words coded
Is it zero ?
Yes
Set coded value of descriptor to zero
NO
Form the "Exclusive OR" sum of all coded words for this Term
Is the net result zero ?
Yes
A5
NO
ADD NUMBER of DISTINCT WORDS IN THIS TERM ("WØ")
ADD EXCLUSION VALUE ("CODE")
CODED
STORE RESULT AS coded value of the descriptor
STORE "ADD1" (Profile Number) IN 2nd WORD of output ITEM
STORE SUM of ADD2+ADD3+ADD5 IN 3rd WORD of OUTPUT ITEM (TYPE, Col, TI, AND SERIAL #)
STORE "ADD7" (LENGTH of ITEM = 4, 14, OR 24) IN 4th WORD of OUTPUT ITEM
Compute Address of 5th WORD of output ITEM AND SAVE IT
"WØ" = 1 ?
NO
SET "K" = K+1
A4
INCREMENT SERIAL Number (ADD6 = ADD6+1) AND SET K = 1

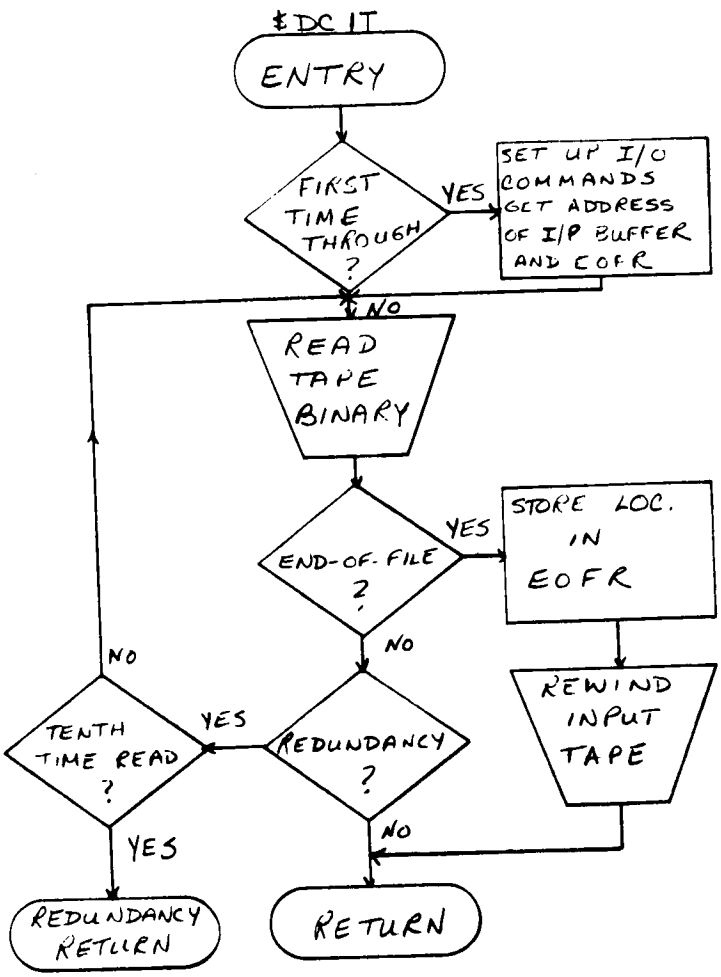

A4
"XTRA" = 0 ?
NO
NEXT2
SET bits 14, 15 EACH TO "1" IN WORD 3 of FIRST OUTPUT ITEM
YES
WHAT IS "TYPE"
RESTORE ALL INDEX REGISTERS
RETURN
TYPE1
TYPE2
TYPE3
TYPE4
IS Col. 71 ("ADDR") A "D" ?
YES
SET "K" = 1
NEXT2
NO
IS IT AN "E" ?
NO
YES
SET "K" = 1
SET INDICATORS for bit 13 ≠ 0
INSERT 3RD WORD INDICATORS INTO bits 13-15 of 2ND THRU KTH OUTPUT ITEMS
NEXT
INSERT 3RD WORD INDICATORS INTO bits 13-15 of 1ST OUTPUT ITEM


TYPE1
TYPE2
"WØ" ≥ 2
Yes
NO
SET INDICATORS TO ADD ALL CODES TO the PROFILE AND TO DICTIONARY
NEXT
SET INDICATORS TO ADD CODED PHRASE TO PROFILE only
SET INDICATORS TO ADD EACH PHRASE WORD TO PROFILE AND TO DICTIONARY
A6
CHAR
NO
"XTRA" = 0
GOOF
YES
A5
SET "K" = 1
STORE PROFILE NUMBER (ADD1) IN THE 2ND WORD of THE SINGLE OUTPUT ITEM
PLACE TYPE (ADD2) AND Col. 71 (ADD3) IN 3RD WORD of OUTPUT ITEM. SET BITS 18-35 All NON-ZERO.
SET "AWFUL" < 0
PLACE ITEM LENGTH (ADD7 = 14 OR 24) IN 4TH WORD of OUTPUT ITEM
SAVE1


NEW
"KEY2" = 0 ?
YES
NO
SET "KEY2" ≠ 0
"XTRA" = 0
YES
NO
CHAR
A7
Form "Exclusive OR" sum of all ("W") 5-character ALPHANUMERIC GROUPS
"XTRA" = 0
NO
YES
INCREMENT COUNT of Processed ALPHANUMERIC WORDS
"W" > 1
YES
NO
SET "AWFUL" > 0
SAVE 1
IS The coded WORD A Forbidden PREPOSITION ?
NO
Yes
Set Bits 30-32 To "001", Thus INDICATING Code of single WORD
Set Exclusion Code (bits 33-65) To zero, Thus Forming A "MAY" Term
SAVE ADDRESS of word 3 of This OUTPUT ITEM
STORE Profile NUMBER ("ADD1") IN 2nd WORD of This OUTPUT ITEM
STORE "ADD2", "ADD3", & "ADD4" (TYPE, col. 71, & SERIAL NUMBER) IN 3rd OUTPUT WORD
INCREMENT SERIAL NUMBER "ADD6" = ADD6+1
form ADDRESS AT which The NEXT OUTPUT ITEM will bEGIN
Set "CHAR2" = 0 "W" = 1
SET INDEX REGISTERS TO PROCESS MORE ALPHANUMERIC DATA
IS ANY MORE DATA TO BE Processed ?
Yes
CHAR
NO
GATHER

Subroutine DCIT

DCIT reads the sorted binary tape B containing the document descriptors. It is entered from subroutine PDTB. The read instructions are set up in its first entry. Upon reading EOF, the tape is rewound.

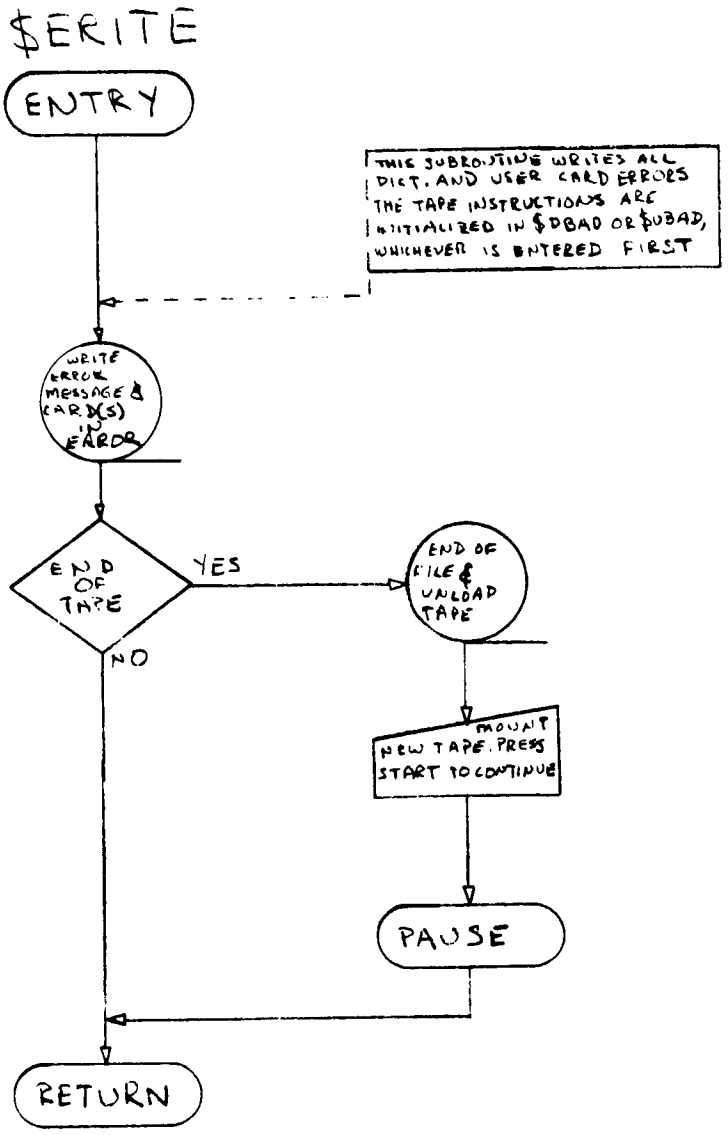

$DC IT
ENTRY
FIRST TIME THROUGH ?
YES
SET UP I/O COMMANDS GET ADDRESS OF I/P BUFFER AND EOFR
NO
READ TAPE BINARY
END-OF-FILE ?
YES
STORE LOC. IN EOFR
REWIND INPUT TAPE
NO
REDUNDANCY ?
YES
TENTH TIME READ ?
NO
YES
REDUNDANCY RETURN
NO
RETURN

Subroutine ERITE

When an illegal card (user or vocabulary) is encountered, one of two subroutines (UBAD or DBAD) will TSX $ERITE, 4 to write the illegal card and the appropriate error message. The first time either $UBAD or $DBAD is entered, it will set up the output instructions in ERITE to write on the system output tape that has been specified in the tape assignment control card. Each time $UBAD or $DBAD is about to go to ERITE it will set up the RCH instruction in ERITE to write the illegal card or error message desired. After ERITE has written the BCD output record, it will return to $UBAD or $DBAD, whichever it came from, which in turn will go back to the subroutine processing user or vocabulary cards.

If an EOT is reached, an EOF will be written and the tape will be unloaded. The following message will be printed on line:

END OF TAPE WRITING cn. MOUNT NEW TAPE-PRESS START.
(c = channel, n = logical tape number)


$ERITE
ENTRY
THIS SUBROUTINE WRITES ALL DICT. AND USER CARD ERRORS THE TAPE INSTRUCTIONS ARE INITIALIZED IN $DBAD OR $UBAD, WHICHEVER IS ENTERED FIRST
WRITE ERROR MESSAGE & CARD(S) IN ERROR
END OF TAPE
YES
NO
END OF FILE & UNLOAD TAPE
MOUNT NEW TAPE. PRESS START TO CONTINUE
PAUSE
RETURN

Subroutine GDNRY

GDNRY processes all vocabulary input cards until a card with ENDICT in cols. 1-6 is encountered. Each card is picked up with a TSX $RDIN, 4. RDIN is a subroutine that reads one card from the input tape into area DCARD-DCARD+11.

Vocabulary input may be in one-card or two-card groups. If one card, col. 72 will be blank; if two, col. 72 of card 1 will contain a C and col. 72 of card 2 will be blank.

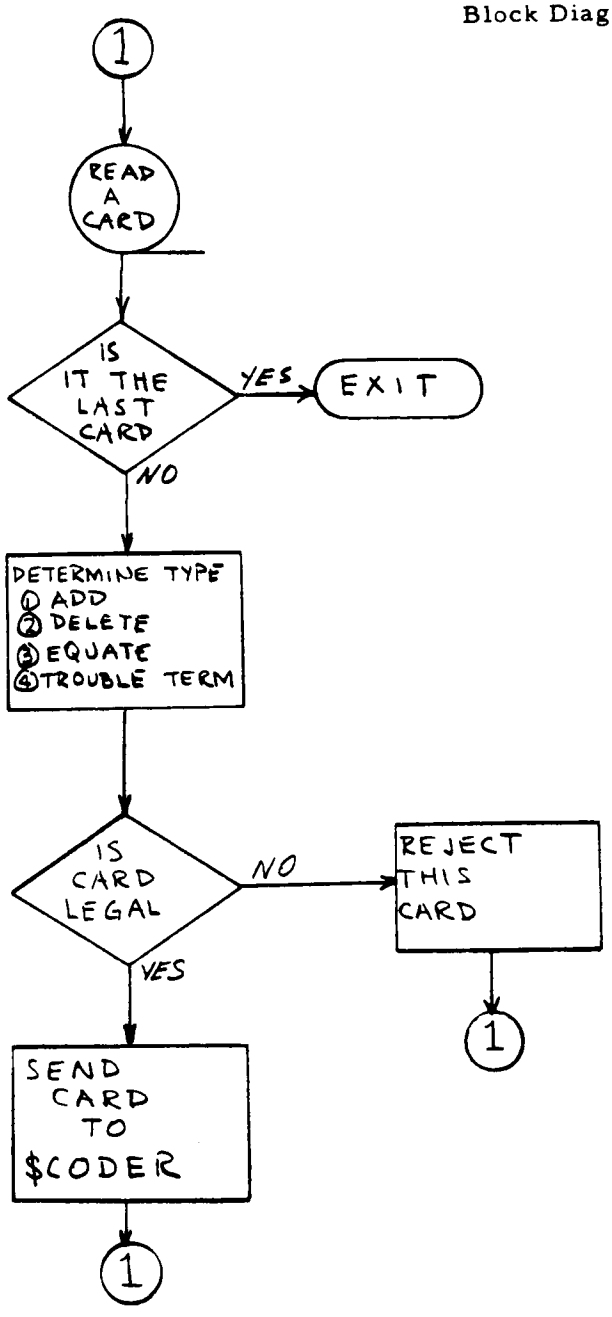
The legal cards with their punches in col. 71 are: add (A), delete (D), equate (E), and trouble term (*). If col. 71 does not contain any of the above, the entire card with an accompanying error message is written on the system output tape by subroutine DBAD. The card following a card with a C in col. 72 must match in col. 71, and in addition be blank in col. 72. Any irregularity will cause a rejection of the first card.

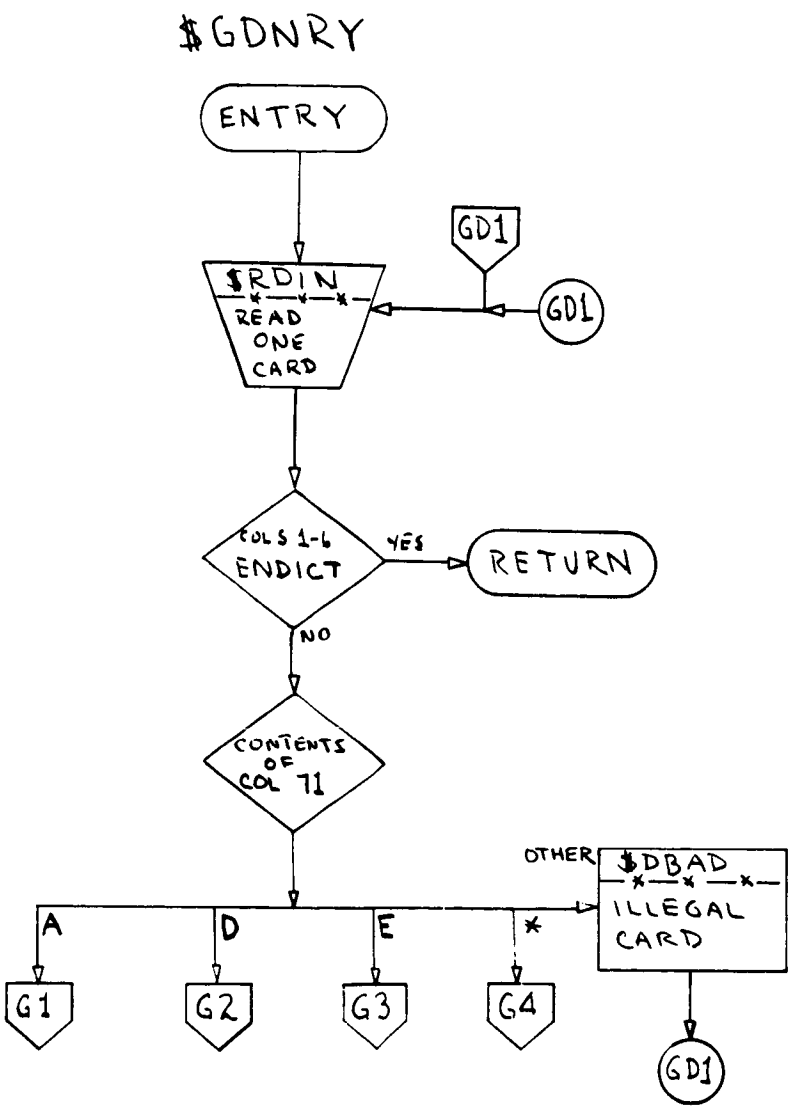
If GDNRY encounters an E-card, it must find a card with a T in col. 71 immediately after it. This contains the primary descriptor to which the phrase on the E-card will be equated. Any violation will cause rejection of the E-card or the E- and T-cards, depending upon the error.

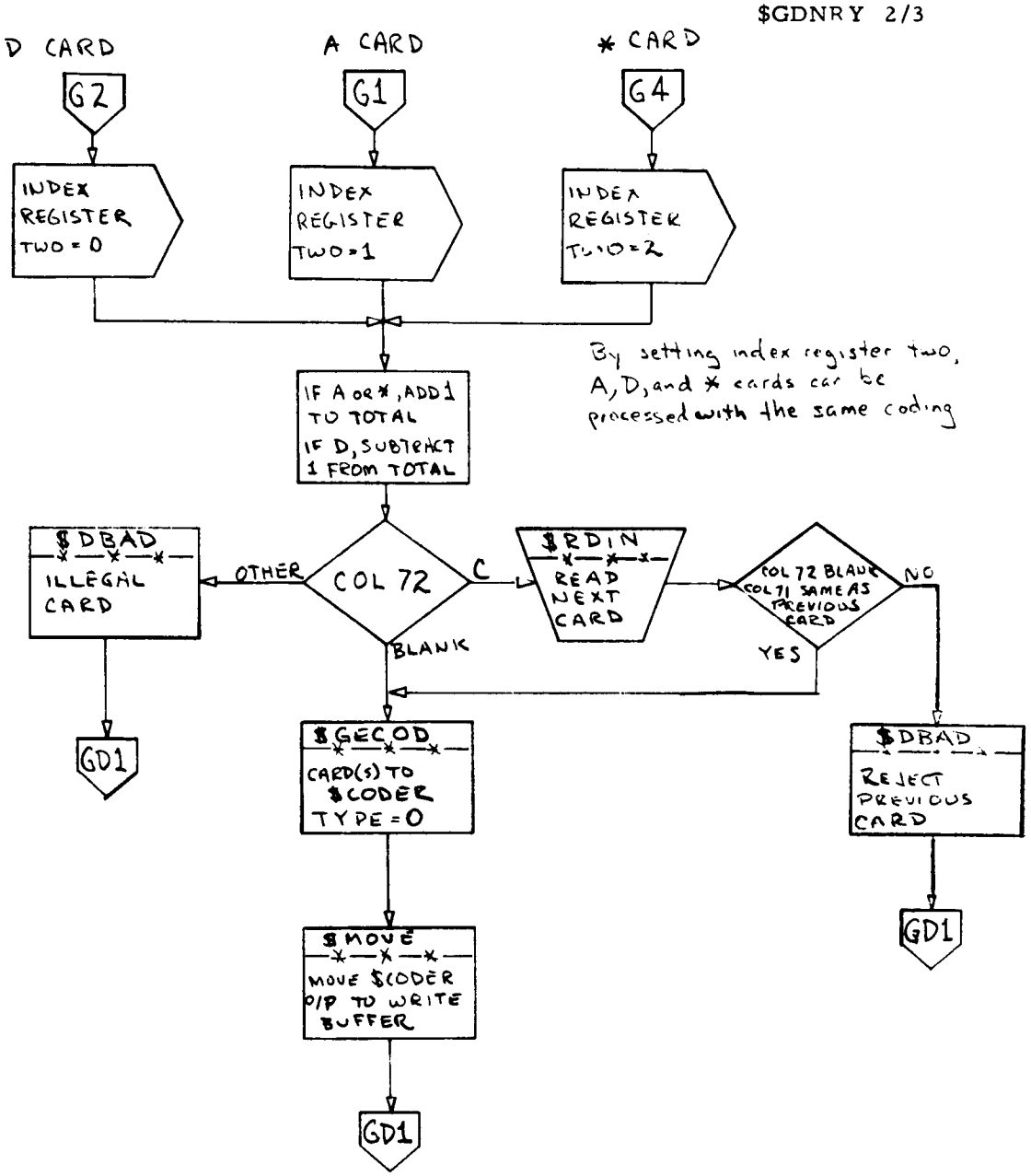
Whenever a valid card group is read in, it will be sent to $CODER.

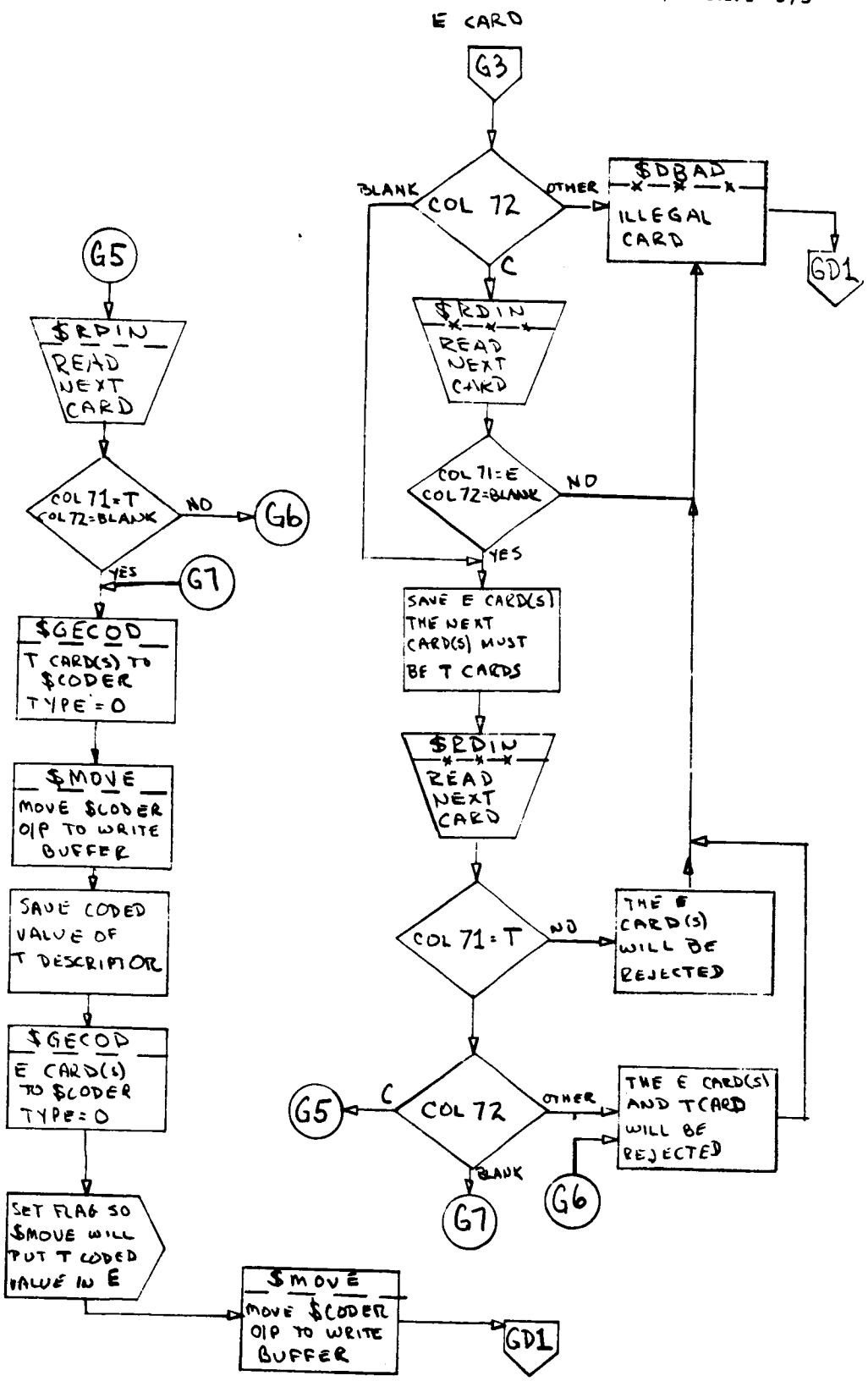
$GDNRY

Block Diagram




$GDNRY
ENTRY
GD1
$RDIN
READ ONE CARD
GD1
COLS 1-6 ENDICT
YES
RETURN
NO
CONTENTS OF COL 71
OTHER
$DBAD
ILLEGAL CARD
A
D
E
*
G1
G2
G3
G4
GD1


D CARD
A CARD
* CARD
G2
G1
G4
INDEX REGISTER TWO = 0
INDEX REGISTER TWO = 1
INDEX REGISTER TWO = 2
By setting index register two, A, D, and * cards can be processed with the same coding
IF A or *, ADD 1 TO TOTAL
IF D, SUBTRACT 1 FROM TOTAL
$DBAD
ILLEGAL CARD
OTHER
COL 72
C
$RDIN
READ NEXT CARD
COL 72 BLANK COL 71 SAME AS PREVIOUS CARD
NO
YES
BLANK
GD1
$GECOD
CARD(S) TO $CODER TYPE = 0
$DBAD
REJECT PREVIOUS CARD
GD1
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
GD1


E CARD
G3
COL 72
BLANK
OTHER
$DBAD
ILLEGAL CARD
GD1
C
$RDIN
READ NEXT CARD
COL 71=E
COL 72=BLANK
NO
YES
SAVE E CARD(S) THE NEXT CARD(S) MUST BE T CARDS
$RDIN
READ NEXT CARD
COL 71=T
NO
THE E CARD(S) WILL BE REJECTED
COL 72
C
G5
OTHER
THE E CARD(S) AND T CARD WILL BE REJECTED
G6
BLANK
G7
G5
$RDIN
READ NEXT CARD
COL 71=T
COL 72=BLANK
NO
G6
YES
G7
$GECOD
T CARD(S) TO $CODER TYPE=0
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
SAVE CODED VALUE OF T DESCRIPTOR
$GECOD
E CARD(S) TO $CODER TYPE=0
SET FLAG SO $MOVE WILL PUT T CODED VALUE IN E
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
GD1

Subroutine GECOD

Whenever a user, vocabulary, or document descriptor is to be sent to $CODER, a TSX $GECOD, 4 is executed. The calling sequence contains the location of the descriptor and the type. GECOD picks these up, sets them in the calling sequence $CODER expects, and then goes to $CODER.


$GECOD
ENTRY
PICK UP TYPE AND LOCATION OF INPUT BUFFER
$CODER
CODE DESCRIPTOR
RETURN

Subroutine GTDOC

This subroutine reads the input document tape, and extracts the header and descriptors for each document. The header and all descriptors are written on tape A. Tape A also contains all vocabulary and user descriptors.

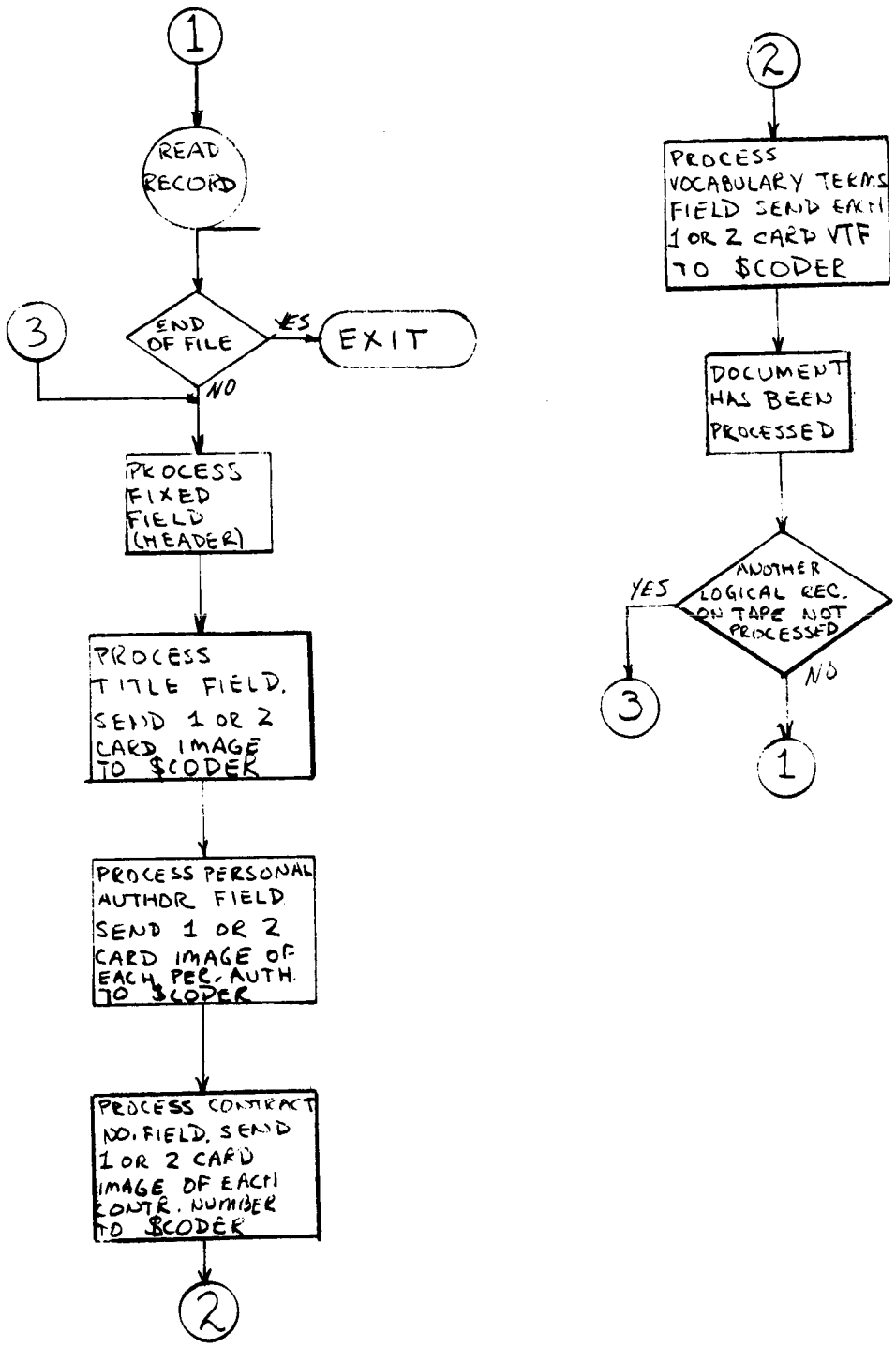
For each document, the following is done: The header is picked up and written on tape A. The next field (title field) is sent to $CODER. Five fields are skipped. The next field (personal author) is processed. Two more fields are skipped. Then the two remaining fields (contract number and descriptor fields) are processed. The last three fields handled by GTDOC may have more than one descriptor; each is separated by a word mark. Any field may be completely empty, in which case the field will contain one dollar sign. The dollar sign is the field separator.

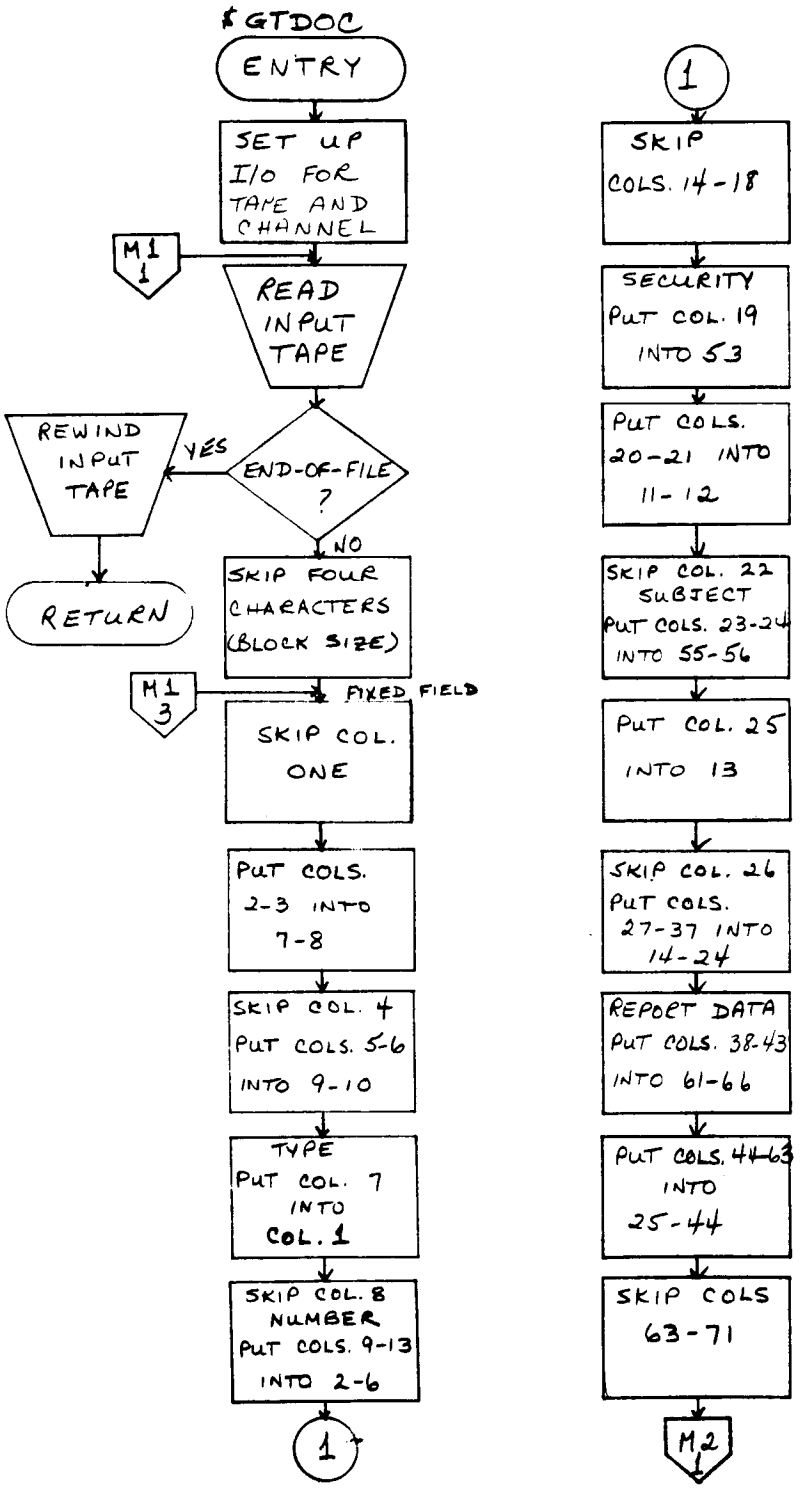
The document field called NOC may contain a price for the document, also indicated by a dollar sign, that is not a field separator. To avoid this false field separator, GTDOC utilizes the fields it wants from the beginning of the logical record, then skips the NOC field by counting backwards from the end of the logical record.

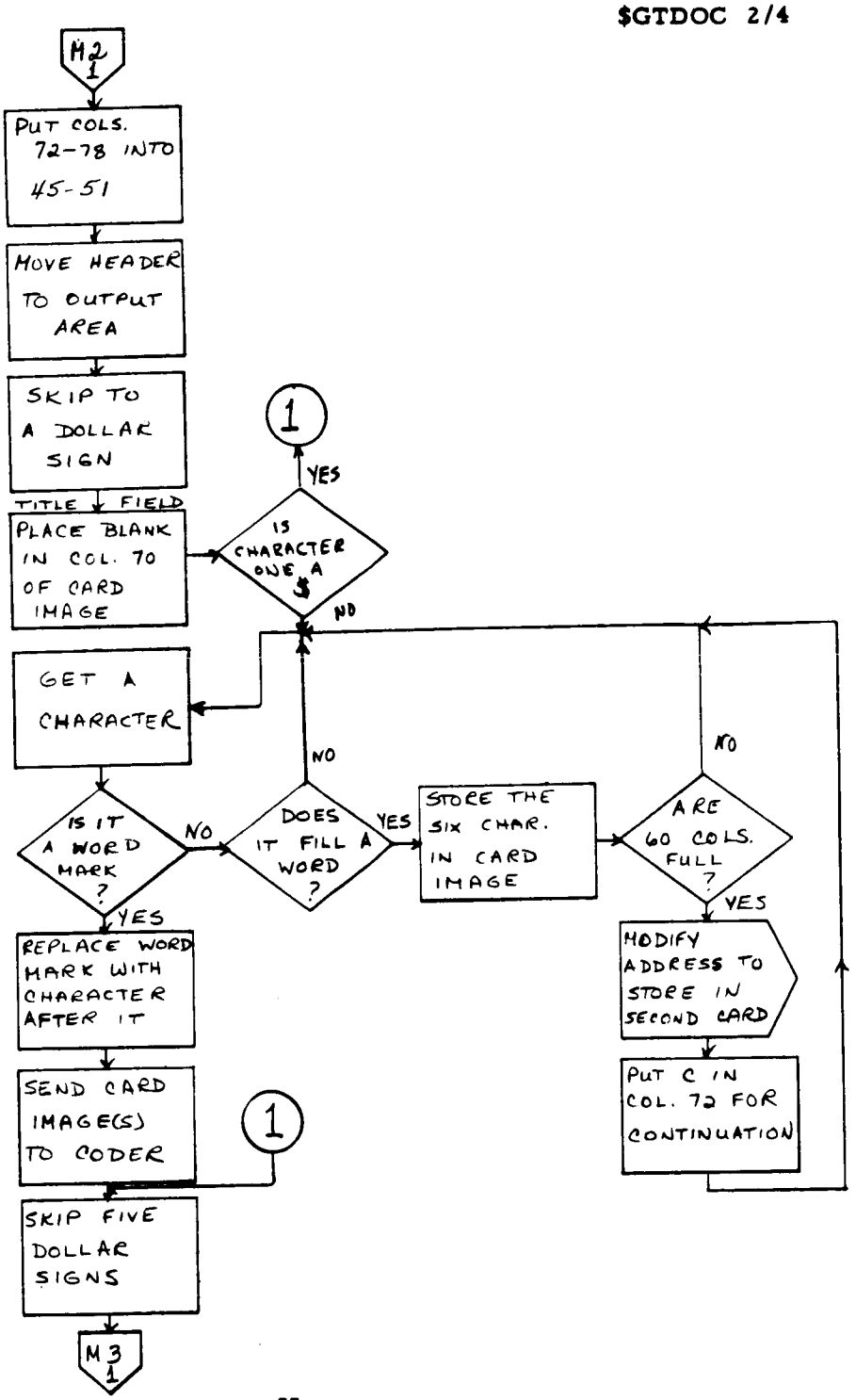
When EOF is encountered in reading the document tape, it is rewound.

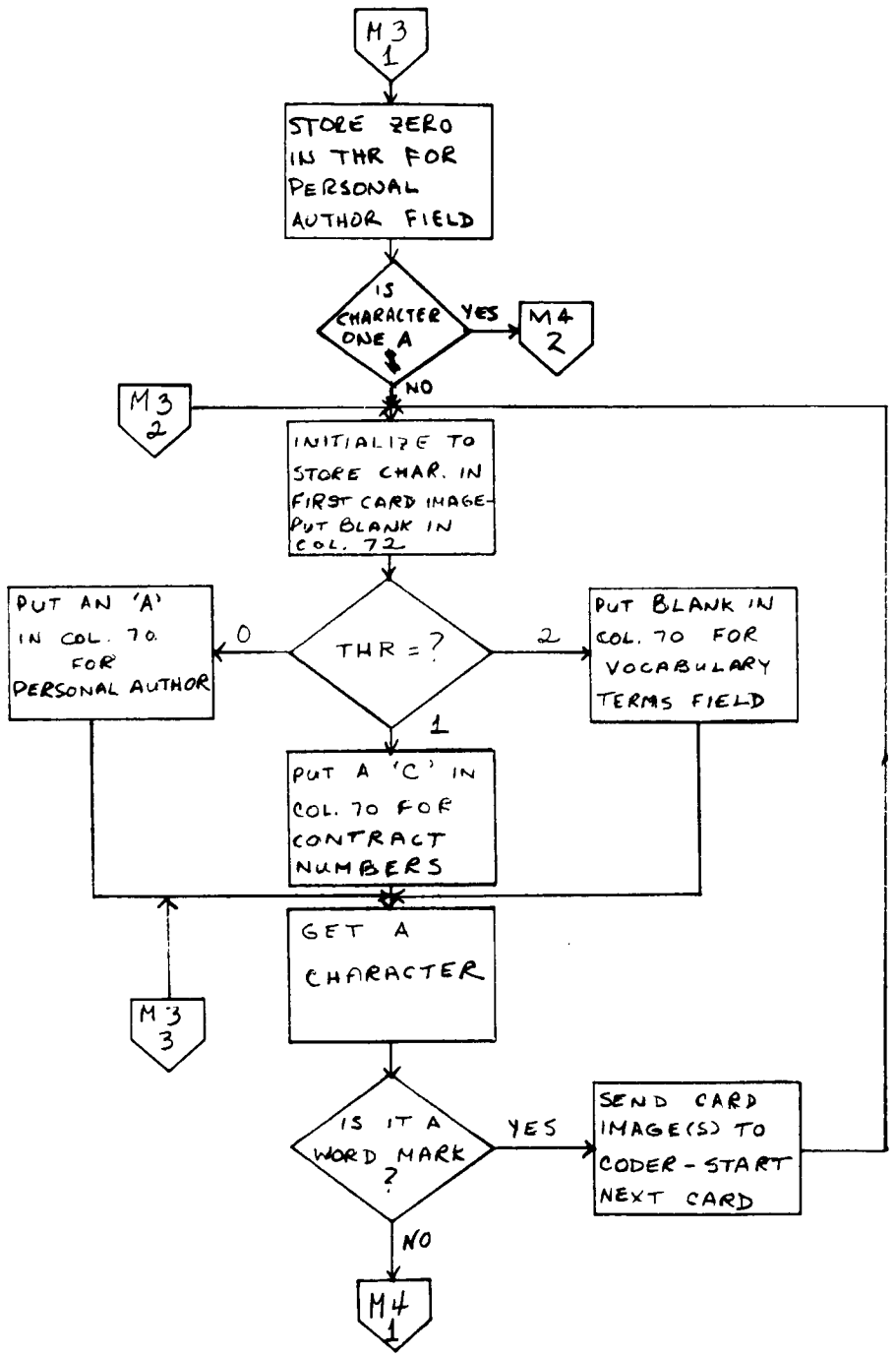
$GTDOC

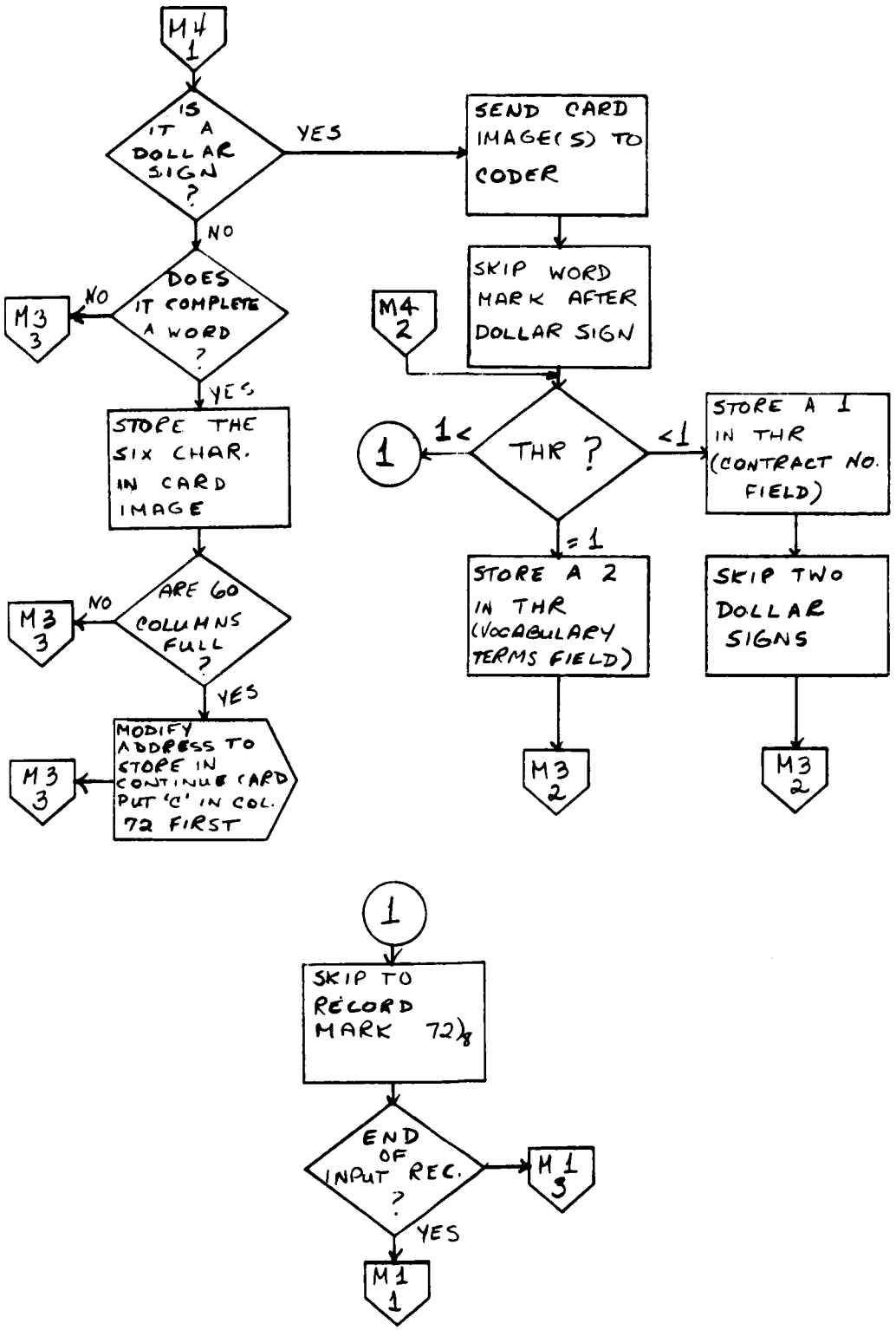
Block Diagram




$GTDOC
ENTRY
SET UP I/O FOR TAPE AND CHANNEL
M1 1
READ INPUT TAPE
END-OF-FILE ?
YES
REWIND INPUT TAPE
RETURN
NO
SKIP FOUR CHARACTERS (BLOCK SIZE)
M1 3
FIXED FIELD
SKIP COL. ONE
PUT COLS. 2-3 INTO 7-8
SKIP COL. 4 PUT COLS. 5-6 INTO 9-10
TYPE PUT COL. 7 INTO COL. 1
SKIP COL. 8 NUMBER PUT COLS. 9-13 INTO 2-6
1
1
SKIP COLS. 14-18
SECURITY PUT COL. 19 INTO 53
PUT COLS. 20-21 INTO 11-12
SKIP COL. 22 SUBJECT PUT COLS. 23-24 INTO 55-56
PUT COL. 25 INTO 13
SKIP COL. 26 PUT COLS. 27-37 INTO 14-24
REPORT DATA PUT COLS. 38-43 INTO 61-66
PUT COLS. 44-63 INTO 25-44
SKIP COLS 63-71
M2 1


M2 1
PUT COLS. 72-78 INTO 45-51
MOVE HEADER TO OUTPUT AREA
SKIP TO A DOLLAR SIGN
TITLE FIELD
PLACE BLANK IN COL. 70 OF CARD IMAGE
1
YES
IS CHARACTER ONE A $
NO
GET A CHARACTER
IS IT A WORD MARK ?
NO
YES
DOES IT FILL A WORD ?
NO
YES
STORE THE SIX CHAR. IN CARD IMAGE
ARE 60 COLS. FULL ?
NO
YES
REPLACE WORD MARK WITH CHARACTER AFTER IT
MODIFY ADDRESS TO STORE IN SECOND CARD
PUT C IN COL. 72 FOR CONTINUATION
SEND CARD IMAGE(S) TO CODER
1
SKIP FIVE DOLLAR SIGNS
M3 1


M3 1
STORE ZERO IN THR FOR PERSONAL AUTHOR FIELD
IS CHARACTER ONE A
YES
M4 2
NO
M3 2
INITIALIZE TO STORE CHAR. IN FIRST CARD IMAGE PUT BLANK IN COL. 72
PUT AN 'A' IN COL. 70 FOR PERSONAL AUTHOR
0
THR = ?
2
PUT BLANK IN COL. 70 FOR VOCABULARY TERMS FIELD
1
PUT A 'C' IN COL. 70 FOR CONTRACT NUMBERS
GET A CHARACTER
M3 3
IS IT A WORD MARK ?
YES
SEND CARD IMAGE(S) TO CODER - START NEXT CARD
NO
M4 1


M4 1
IS IT A DOLLAR SIGN ?
YES
SEND CARD IMAGE(S) TO CODER
NO
DOES IT COMPLETE A WORD ?
NO
M3 3
YES
STORE THE SIX CHAR. IN CARD IMAGE
ARE 60 COLUMNS FULL ?
NO
M3 3
YES
MODIFY ADDRESS TO STORE IN CONTINUE CARD PUT 'C' IN COL. 72 FIRST
M3 3
M4 2
SKIP WORD MARK AFTER DOLLAR SIGN
THR ?
1<
1
<1
STORE A 1 IN THR (CONTRACT NO. FIELD)
SKIP TWO DOLLAR SIGNS
M3 2
=1
STORE A 2 IN THR (VOCABULARY TERMS FIELD)
M3 2
1
SKIP TO RECORD MARK 72
END OF INPUT REC. ?
M1 3
YES
M1 1

Subroutine GTUSE

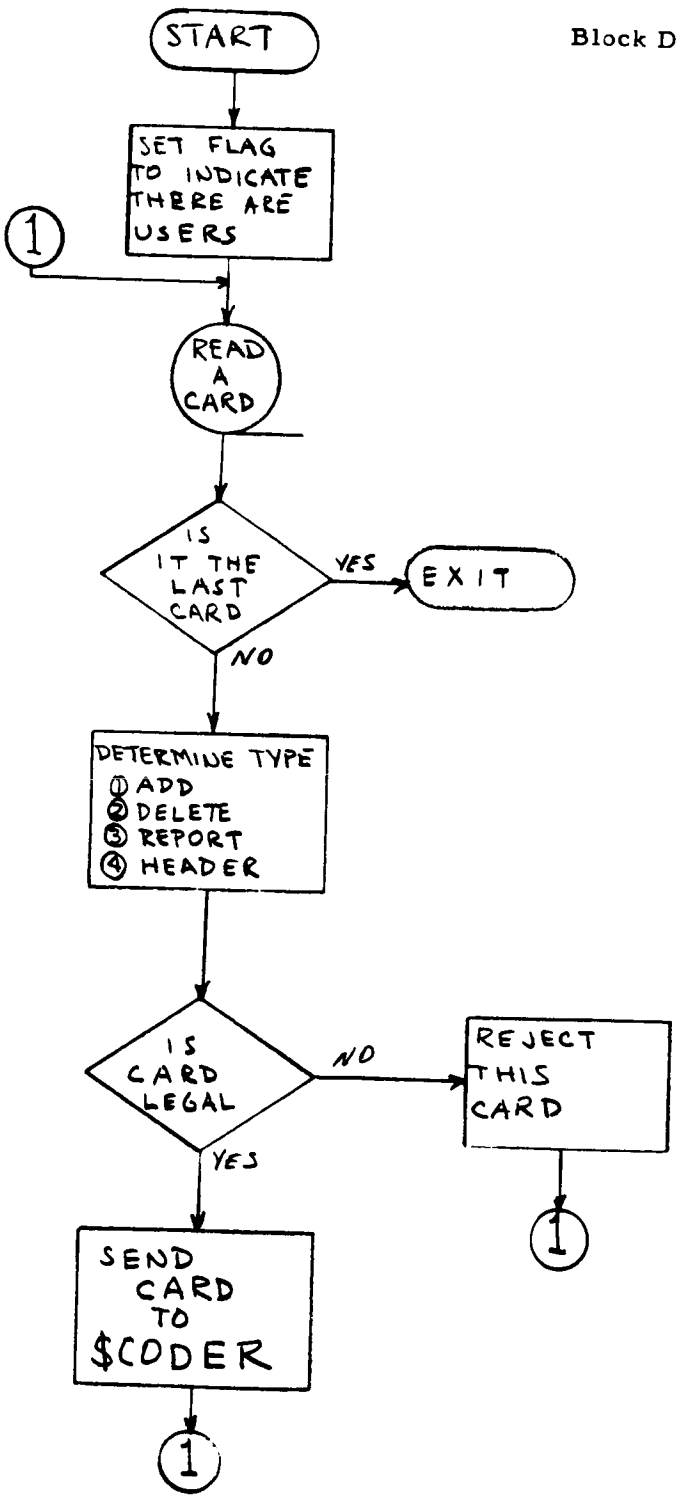
GTUSE processes all user input cards until a card with ENDUSR in cols. 1-6 is encountered. Each card is picked up with a TSX $RDIN, 4. Subroutine RDIN reads one card from the input tape into area DCARD-DCARD+11. The user number (cols. 61-66) is sent to subroutine RBLNK where all blanks are turned to zeros. This eliminates keypunching user numbers with leading zeros. Next, col. 71 is checked to see what kind of user card is being processed. The legal punches in col. 71 are "blank" (new user add), A (old user add), D (delete), R (report), or H (header). If the card does not contain any of the above, the entire card, with an accompanying error message, is written on the system output tape by subroutine UBAD as an illegal card. The next card is then read.

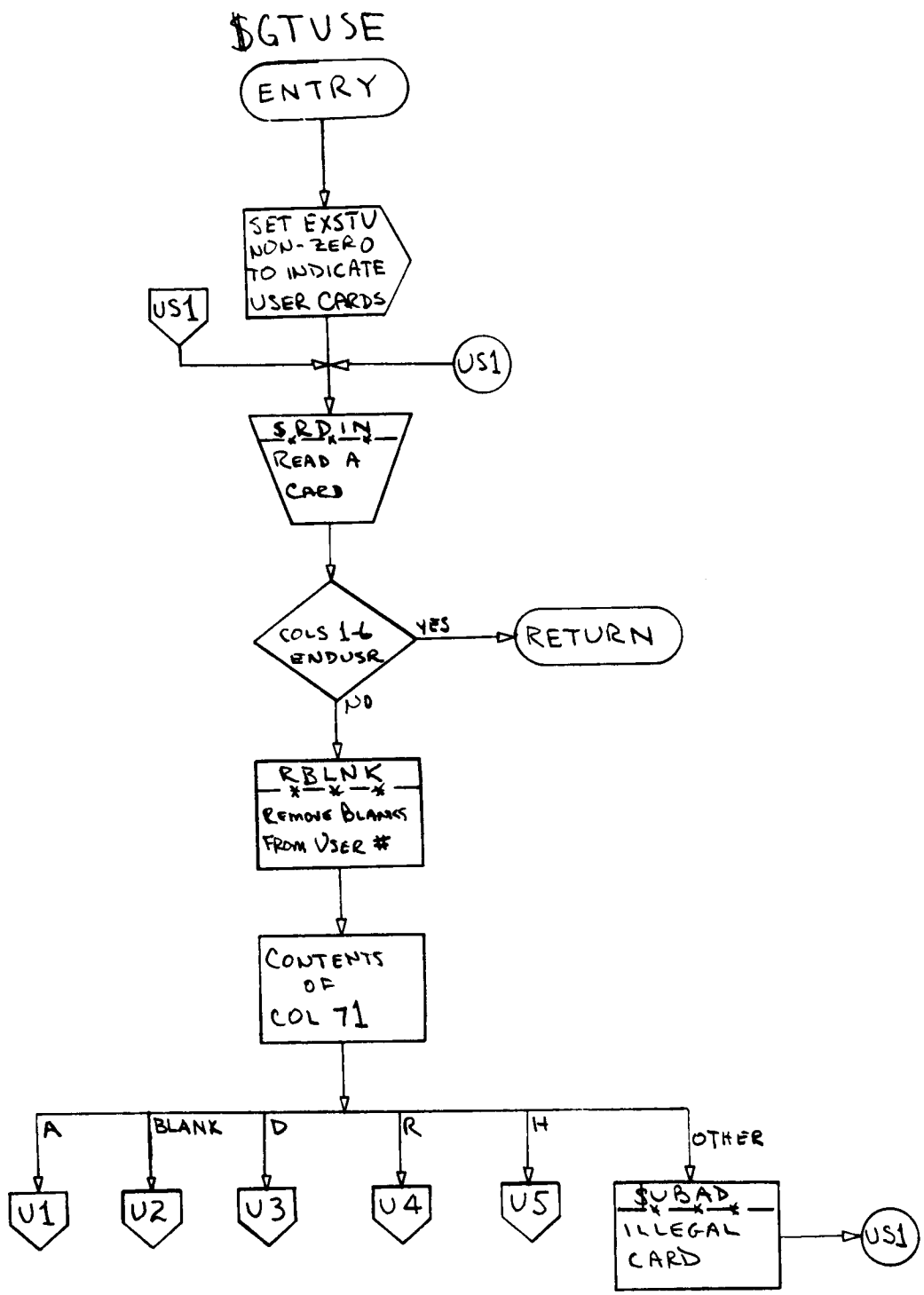
Delete cards may be of two types. One type deletes a user descriptor and the second deletes an entire user profile. The latter is denoted by a U punched in col. 72. With the exception of report cards, user input may be in one-card or two-card groups. If one card, col. 72 will be blank. If two cards, col. 72 of card 1 will contain a C and col. 72 of card 2 will be blank. Any other punch is illegal. In two-card groups, col. 71 of both cards must agree.

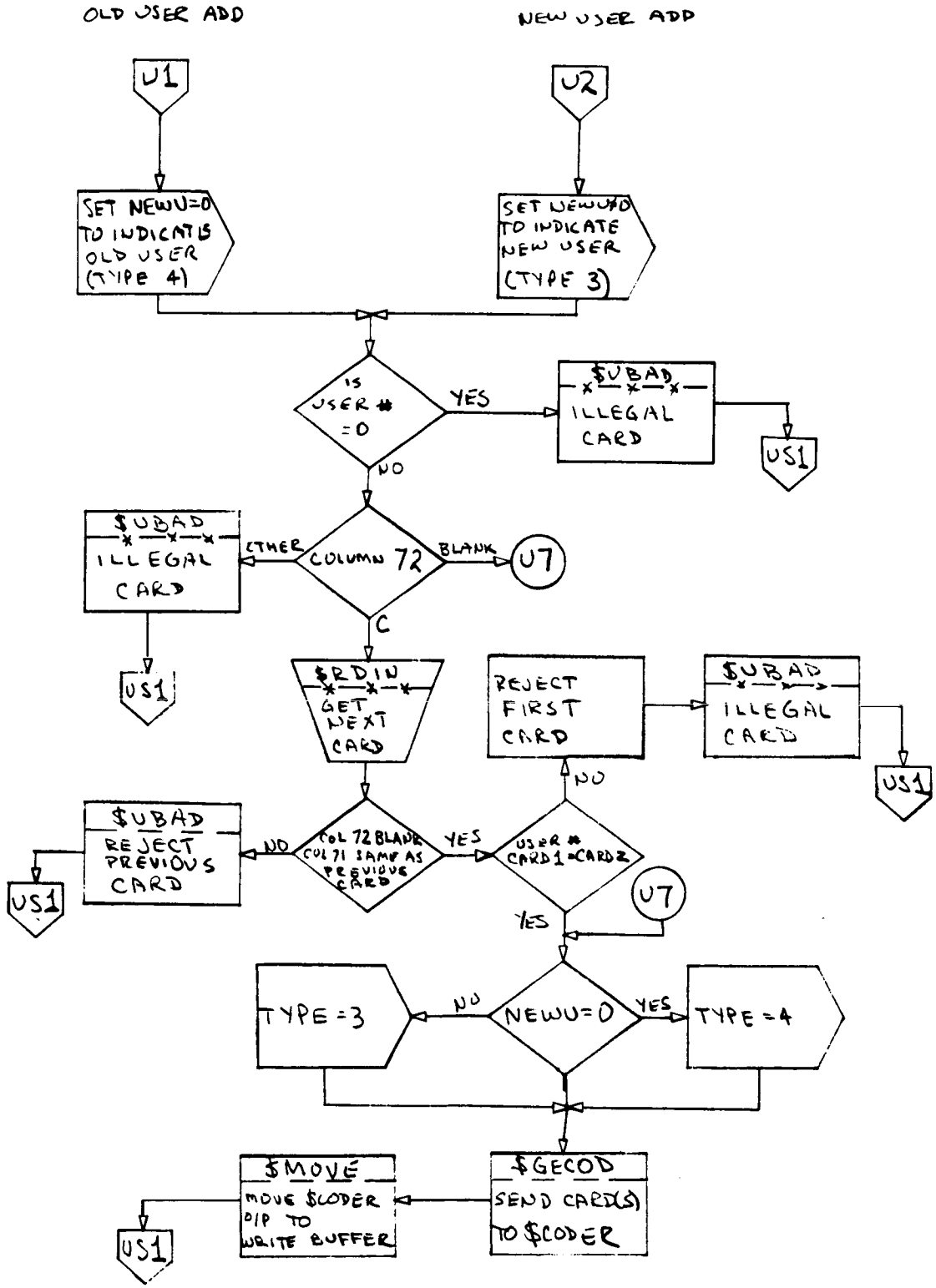
All add and delete cards are sent to $CODER and then written on tape A, using subroutines GECOD, MOVE, and WTAPA. Tape A contains vocabulary and document input, in addition to user input. Report type cards and header cards are not sent to $CODER, but are written directly on tape A by $WTAPA.

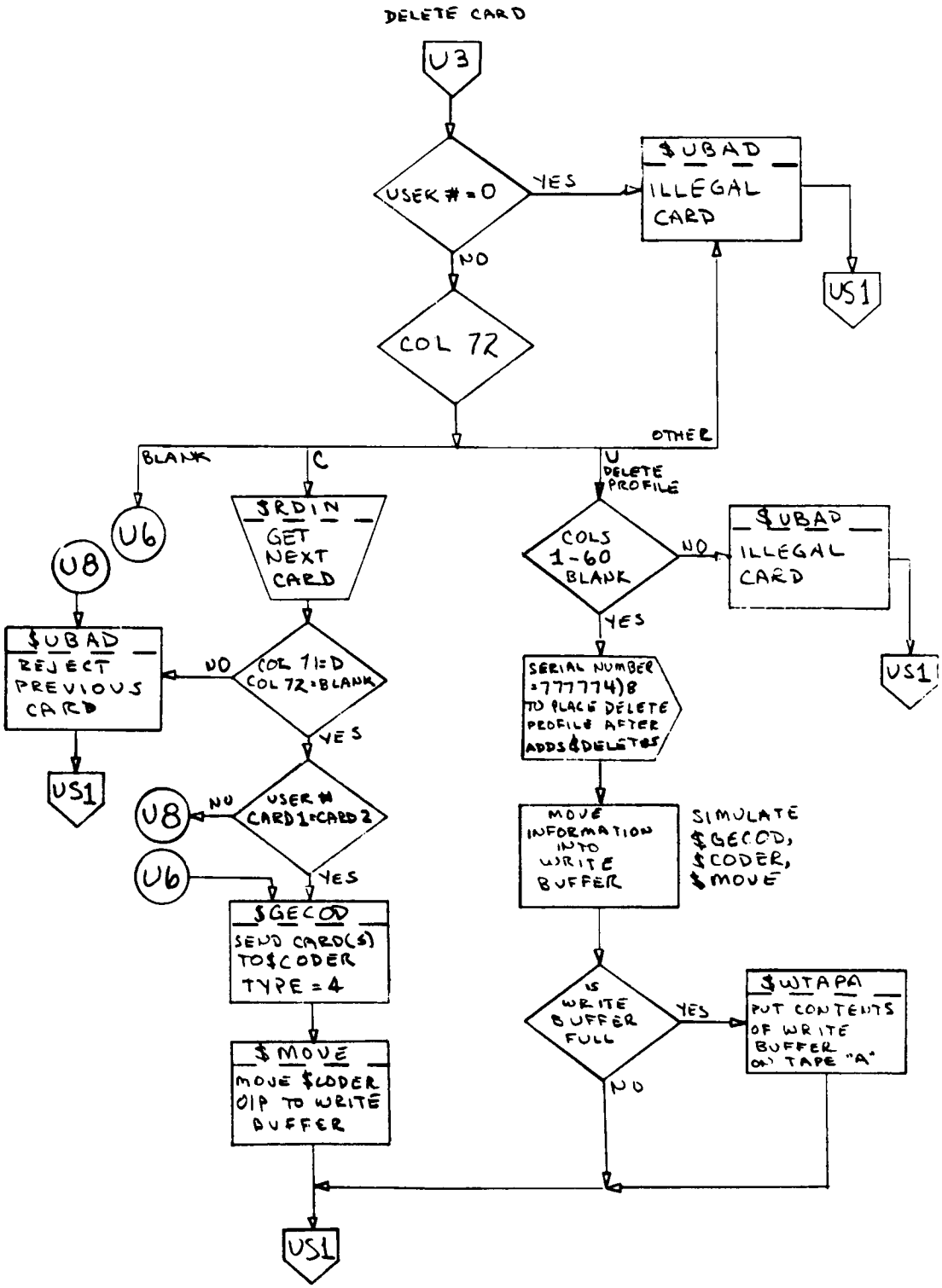
Header cards are given a serialization number of 000000)8 to place them before any descriptors for that user. Delete profile and report cards are given very high numbers to place them after all add and delete descriptor cards. Their serial numbers are 777774)8 and 777776)8, respectively. The serial numbers for adds and deletes are assigned by $CODER, and will always be between 000001)8 and 777773)8.

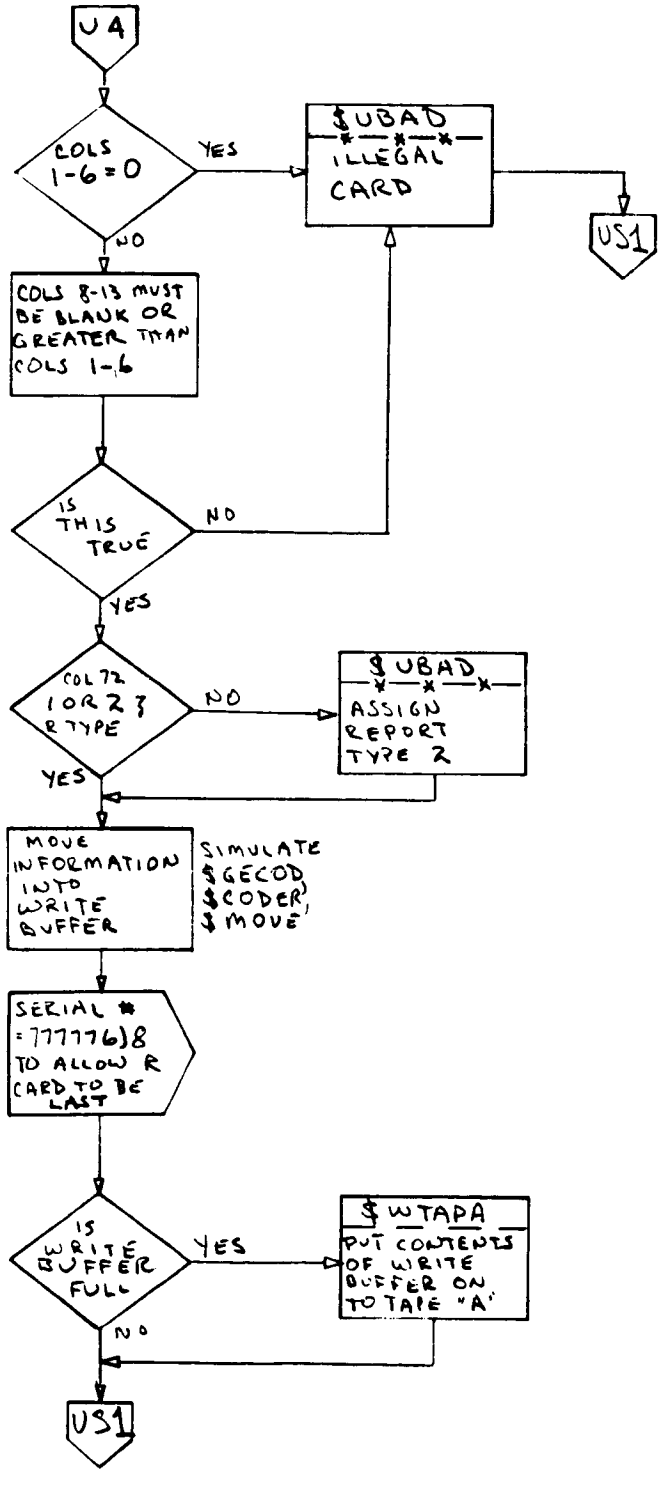
Block Diagram

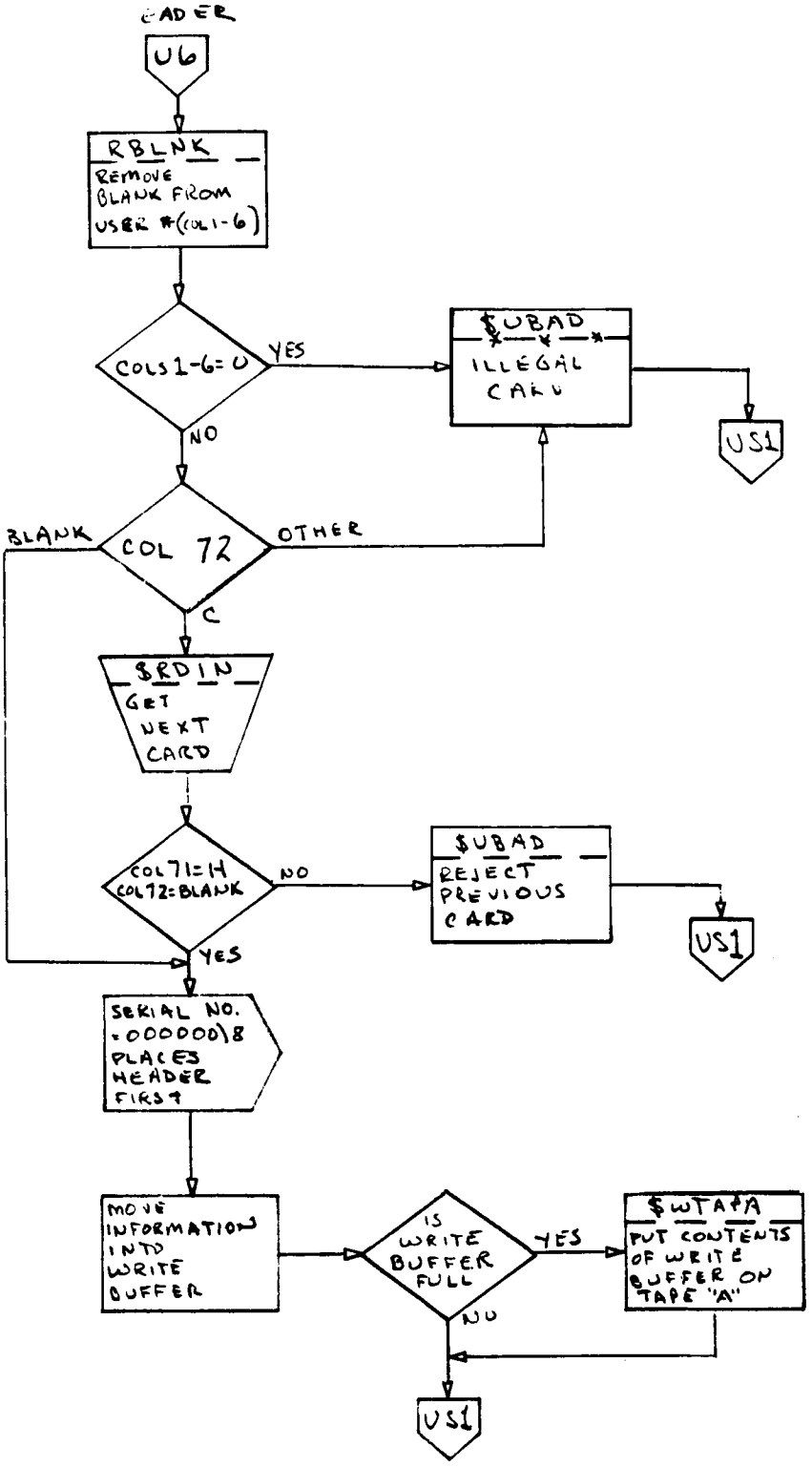



$GTUSE
ENTRY
SET EXSTU NON-ZERO TO INDICATE USER CARDS
US1
US1
$RDIN
READ A CARD
COLS 1-6 ENDUSR
YES
RETURN
NO
RBLNK
REMOVE BLANKS FROM USER #
CONTENTS OF COL 71
A
BLANK
D
R
H
OTHER
U1
U2
U3
U4
U5
$UBAD
ILLEGAL CARD
US1


OLD USER ADD
NEW USER ADD
U1
U2
SET NEWU=0 TO INDICATE OLD USER (TYPE 4)
SET NEWU=0 TO INDICATE NEW USER (TYPE 3)
IS USER # =0
YES
NO
$UBAD ILLEGAL CARD
US1
$UBAD ILLEGAL CARD
EITHER
COLUMN 72
BLANK
U7
C
US1
$RDIN GET NEXT CARD
REJECT FIRST CARD
$UBAD ILLEGAL CARD
US1
NO
$UBAD REJECT PREVIOUS CARD
US1
NO
COL 72 BLANK COL 71 SAME AS PREVIOUS CARD
YES
USER # CARD 1 = CARD 2
U7
YES
TYPE =3
NO
NEWU=0
YES
TYPE =4
$MOVE MOVE $LODER DIP TO WRITE BUFFER
US1
$GECOD SEND CARD(S) TO $LODER


DELETE CARD
U3
USER # = 0
YES
$UBAD
ILLEGAL CARD
US1
NO
COL 72
OTHER
BLANK
C
U
DELETE PROFILE
U6
U8
$RDIN
GET NEXT CARD
COLS 1-60 BLANK
NO
$UBAD
ILLEGAL CARD
US1
YES
$UBAD
REJECT PREVIOUS CARD
US1
NO
COL 1 = D
COL 72 = BLANK
YES
SERIAL NUMBER = 777774)8 TO PLACE DELETE PROFILE AFTER ADDS & DELETES
U8
NO
USER # CARD 1 = CARD 2
YES
U6
MOVE INFORMATION INTO WRITE BUFFER
SIMULATE $GECOD, $CODER, $MOVE
$GECOD
SEND CARD(S) TO $CODER TYPE = 4
IS WRITE BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TAPE "A"
NO
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
US1

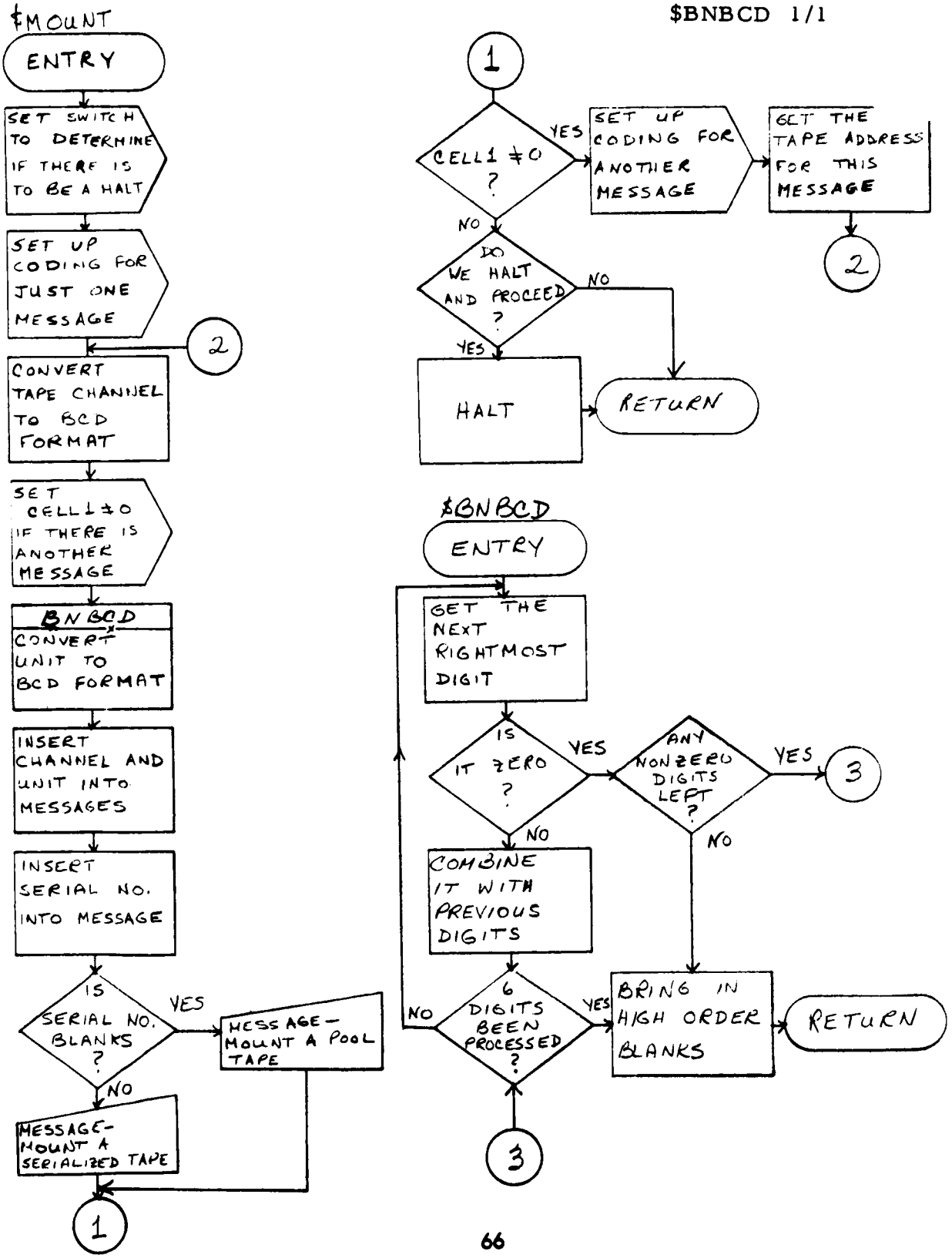

U4
COLS 1-6 = 0
YES
$UBAD
ILLEGAL CARD
US1
NO
COLS 8-13 MUST BE BLANK OR GREATER THAN COLS 1-6
IS THIS TRUE
NO
YES
COL 72 1 OR 2 ? R TYPE
NO
$UBAD
ASSIGN REPORT TYPE 2
YES
MOVE INFORMATION INTO WRITE BUFFER
SIMULATE $GECOD, $CODER, $MOVE
SERIAL # = 777776)8 TO ALLOW R CARD TO BE LAST
IS WRITE BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TO TAPE "A"
NO
US1


CADER
U6
RBLNK
REMOVE BLANK FROM USER #(COL 1-6)
COLS 1-6 = U
YES
$UBAD
ILLEGAL CARD
US1
NO
BLANK
COL 72
OTHER
C
$RDIN
GET NEXT CARD
COL 71 = H
COL 72 = BLANK
NO
$UBAD
REJECT PREVIOUS CARD
US1
YES
SERIAL NO. +000000\8 PLACES HEADER FIRST
MOVE INFORMATION INTO WRITE BUFFER
IS WRITE BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TAPE "A"
NO
US1

Subroutine MOUNT - BNBCD

This subroutine prints messages on the online printer when a tape has to be mounted. If the tape has a serial number, this will be given in the message; if not, the message will ask that a pool tape be mounted. This subroutine has within it a routine to convert an unsigned binary integer to a BCD integer with leading blanks.

$MOUNT 1/1
$BNBCD 1/1



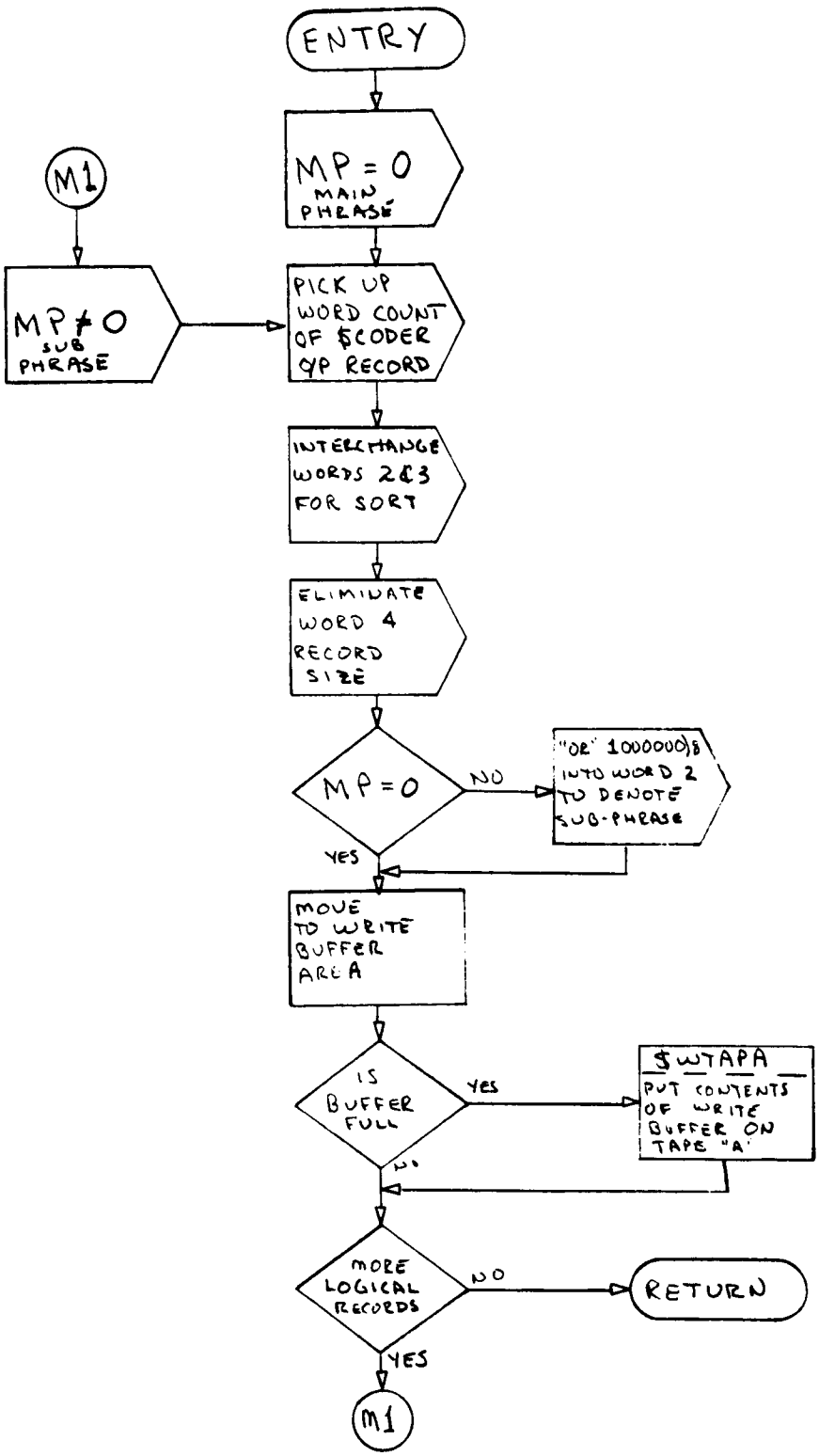
Subroutine MOVE

MOVE is entered after every TSX $CODER, 4. It picks up the logical records put out by $CODER (as few as one, as many as eight) and places them in BLOCK, preparing them to be written on tape A. After each logical record is moved into the BLOCK area, a check is made to see whether or not six logical records are in BLOCK. If so, MOVE will TSX $WTAPA, 4 to write out these six. When it returns it will process the remaining records produced by $CODER.

The first word of every main phrase from $CODER will be masked to contain a "1" in bit 17. All subphrases will be masked so their first words will contain a "0" in this same position.

Special attention is given to vocabulary equate cards. If an E-card has come from $CODER, the coded value of its accompanying T-card will be placed in word 24 of the logical record containing the E-card main phrase.

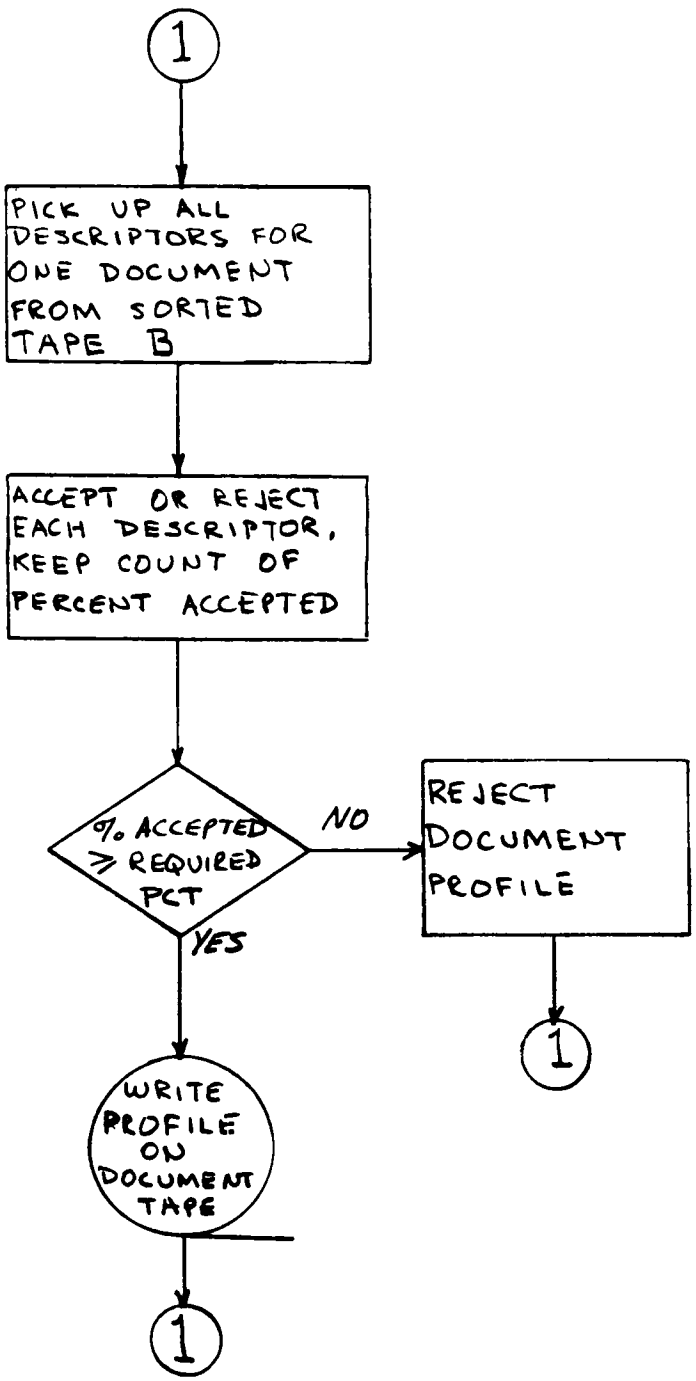
$MOVE


ENTRY
MP = 0
MAIN PHRASE
M1
MP ≠ 0
SUB PHRASE
PICK UP WORD COUNT OF $CODER OP RECORD
INTERCHANGE WORDS 2&3 FOR SORT
ELIMINATE WORD 4 RECORD SIZE
MP = 0
NO
"OR" 1000000)8 INTO WORD 2 TO DENOTE SUB-PHRASE
YES
MOVE TO WRITE BUFFER AREA
IS BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TAPE "A"
NO
MORE LOGICAL RECORDS
NO
RETURN
YES
M1

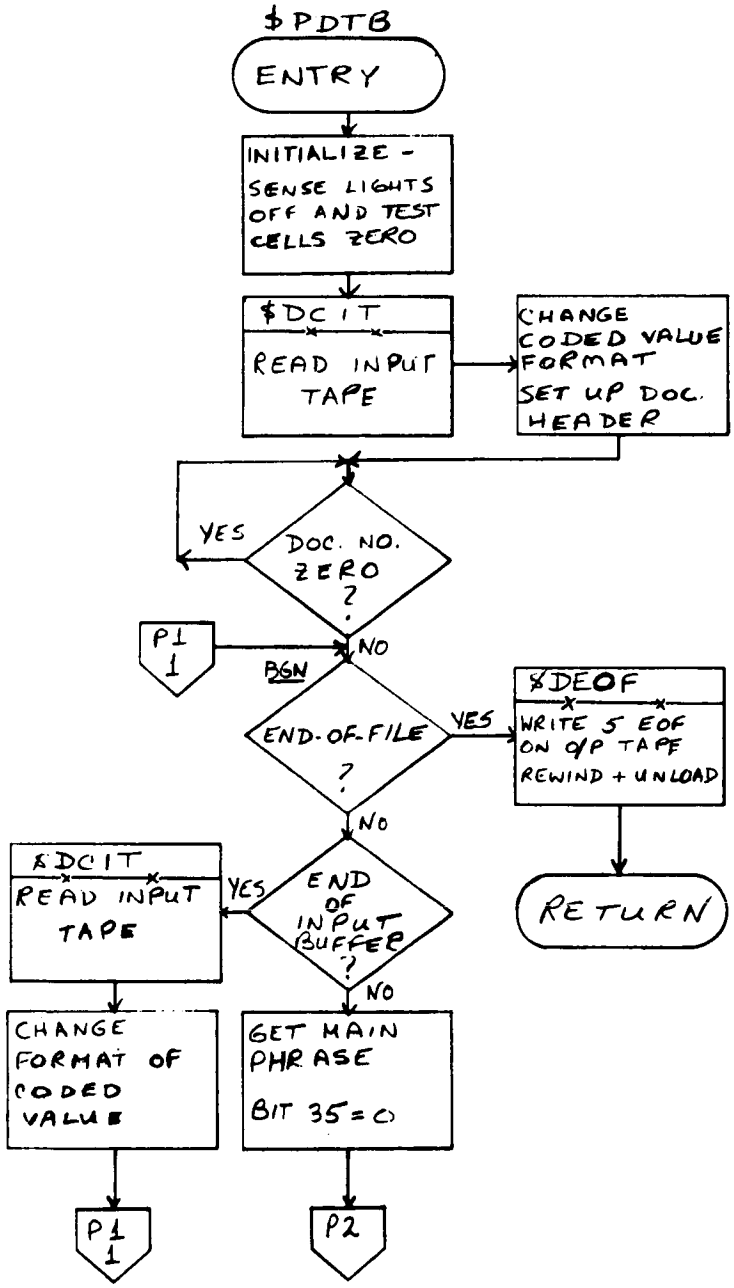
Subroutine PDTB

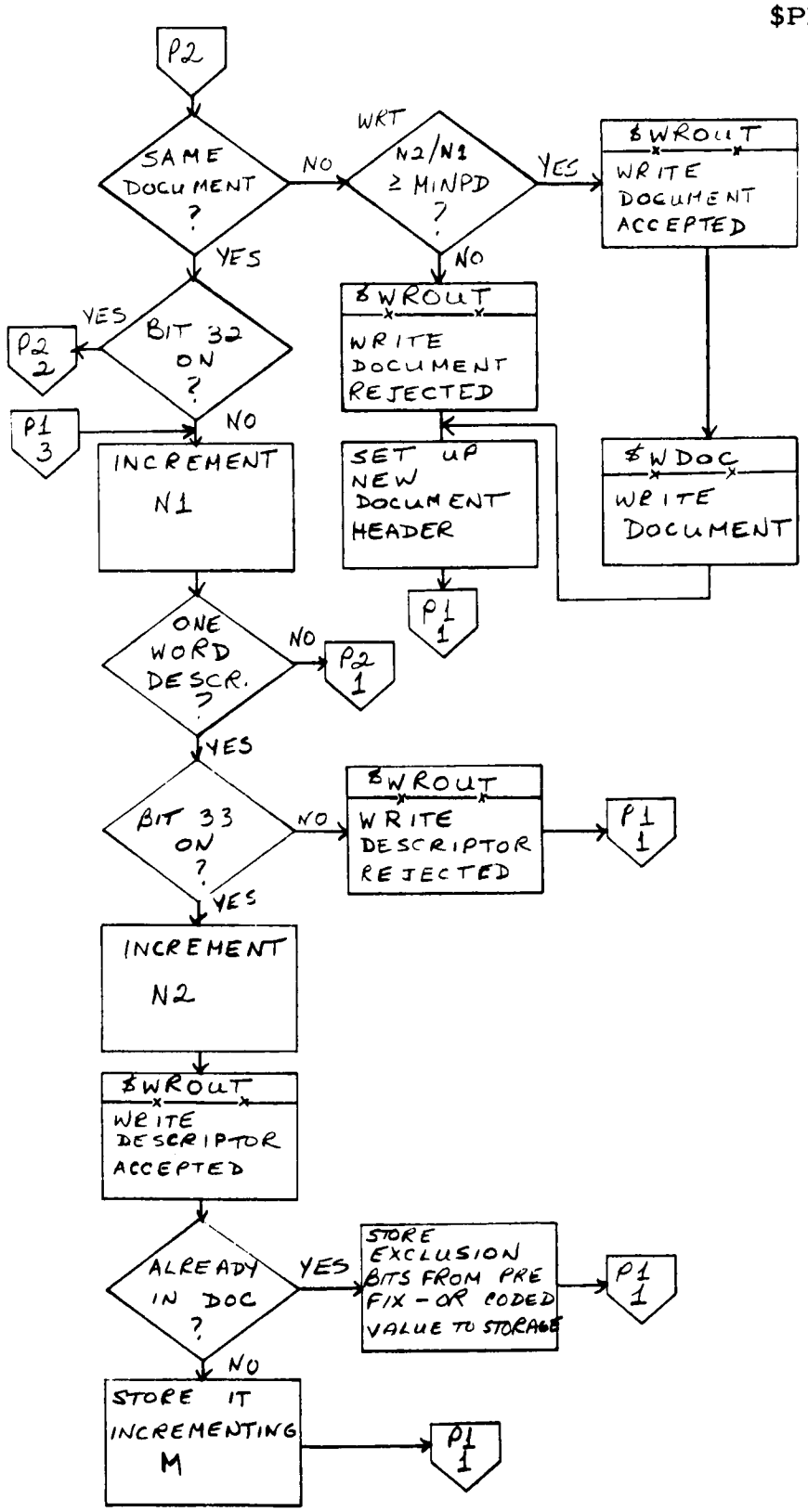
PDTB builds the binary document profile tape. The input tape is sorted tape B. The first descriptor for each document is its header. Following are its descriptors. Each descriptor is checked to see if it should be added to the document profile. When all of one document's descriptors have been processed, the percentage that have been accepted is compared against the control card parameter MIND. If the percent accepted is greater than or equal to MIND, the document is added to the binary tape; otherwise, it is rejected. The acceptance or rejection of every descriptor and document is noted on the system output tape.

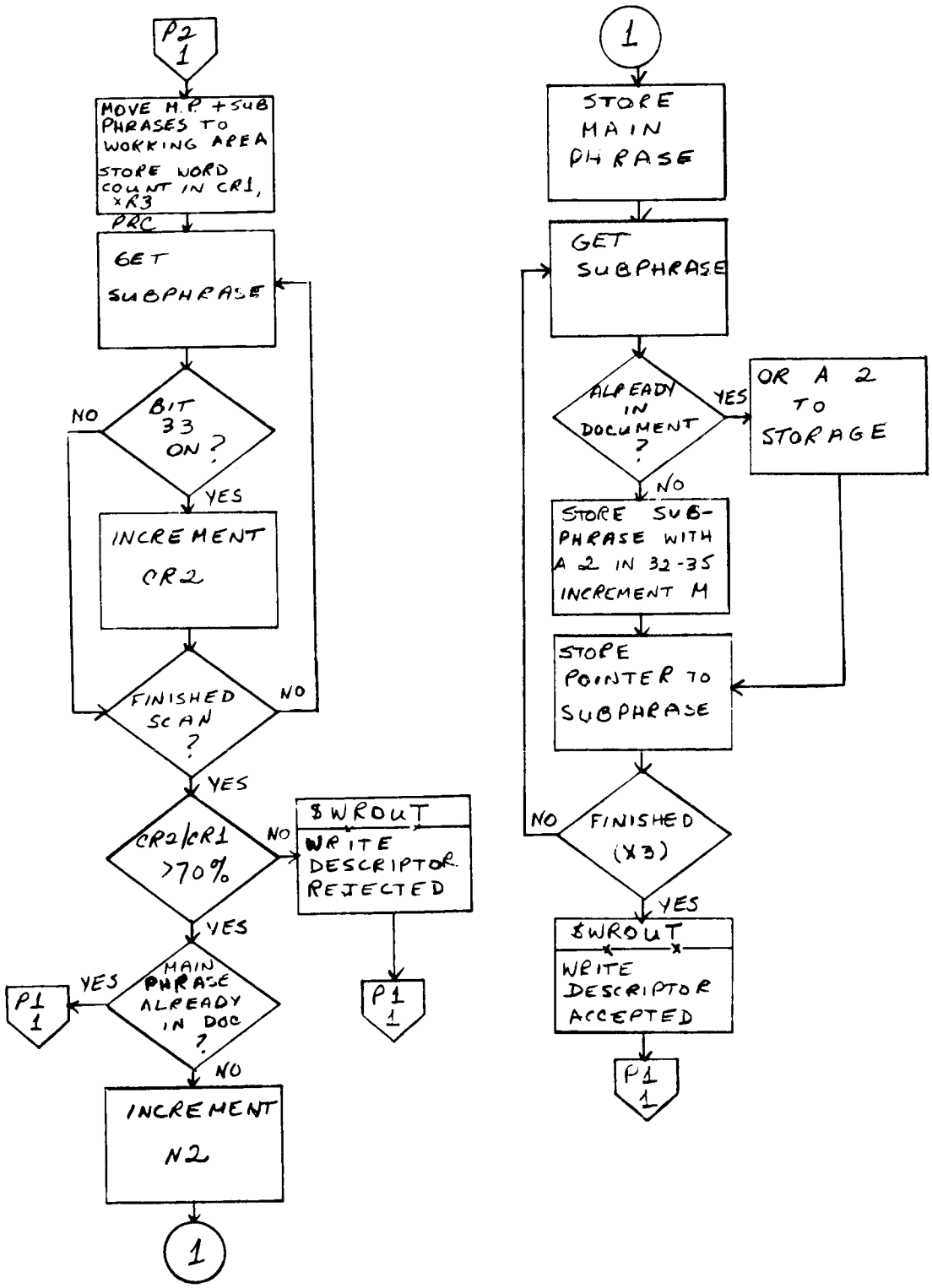
Block Diagram

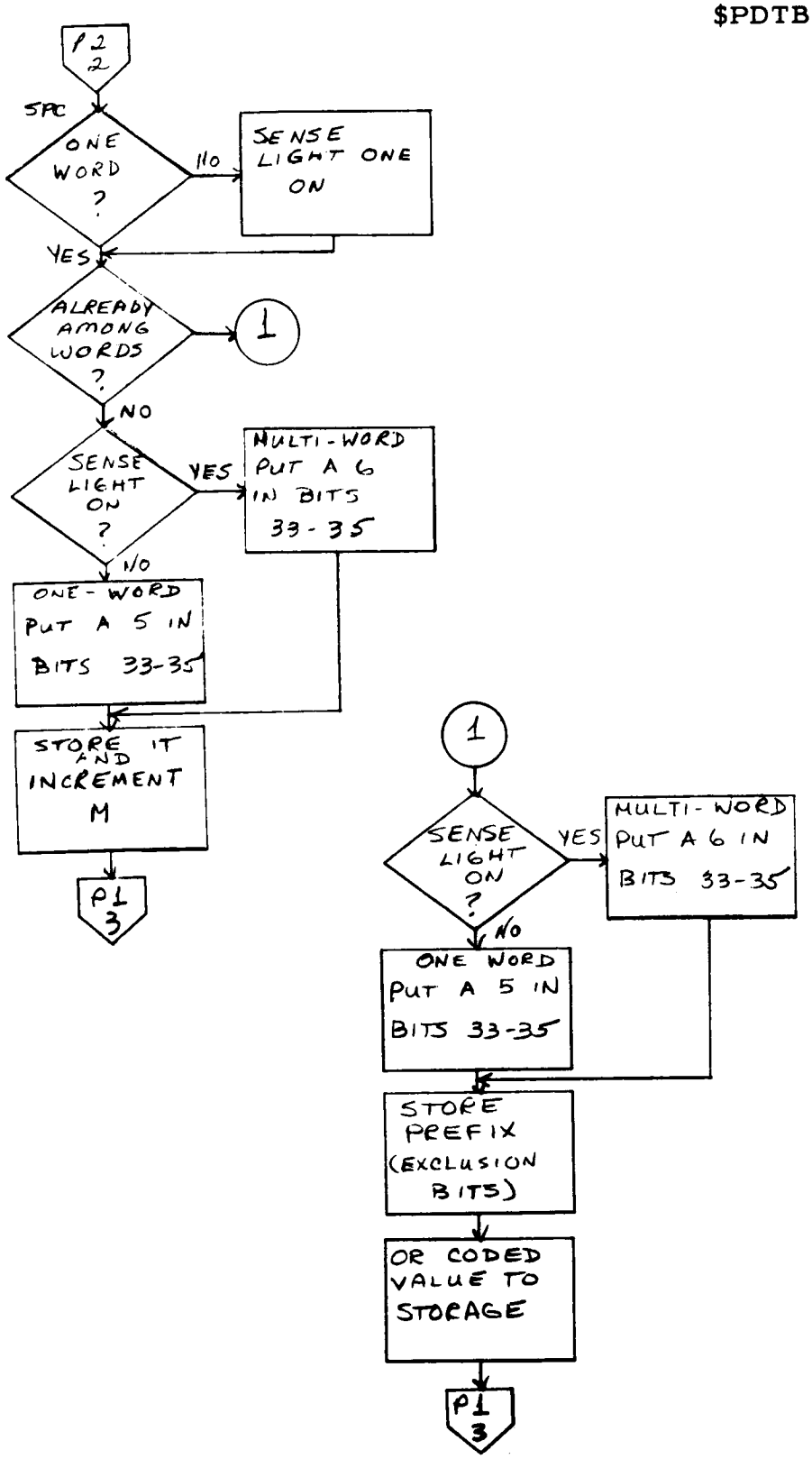


EVERY DESCRIPTOR, WHETHER ACCEPTED OR REJECTED IS SO NOTED ON THE BCD OUTPUT TAPE. THE SAME IS DONE FOR EACH PROFILE


$PDTB
ENTRY
INITIALIZE - SENSE LIGHTS OFF AND TEST CELLS ZERO
$DC IT
READ INPUT TAPE
CHANGE CODED VALUE FORMAT SET UP DOC. HEADER
YES
DOC. NO. ZERO ?
P1 1
NO
BGN
END-OF-FILE ?
YES
$DEOF
WRITE 5 EOF ON O/P TAPE REWIND + UNLOAD
RETURN
NO
$DC IT
READ INPUT TAPE
YES
END OF INPUT BUFFER ?
NO
CHANGE FORMAT OF CODED VALUE
GET MAIN PHRASE
BIT 35 = 0
P1 1
P2

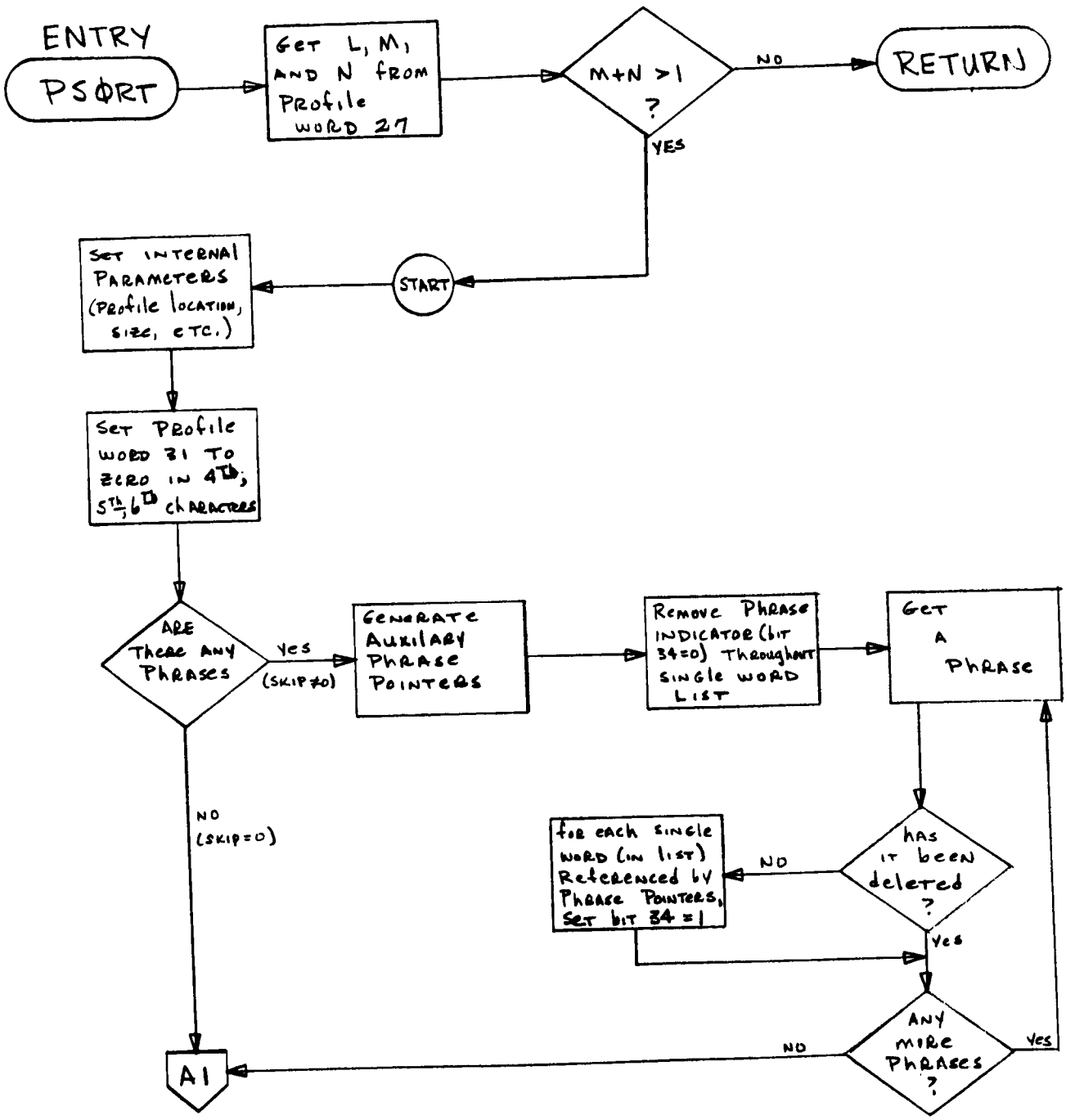

P2
SAME DOCUMENT ?
NO
WRT N2/N1 ≥ MINPD ?
YES
$WROUT WRITE DOCUMENT ACCEPTED
YES
NO
BIT 32 ON ?
YES
P2 2
$WROUT WRITE DOCUMENT REJECTED
P1 3
NO
INCREMENT N1
SET UP NEW DOCUMENT HEADER
$WDOC WRITE DOCUMENT
P1 1
ONE WORD DESCR. ?
NO
P2 1
YES
BIT 33 ON ?
NO
$WROUT WRITE DESCRIPTOR REJECTED
P1 1
YES
INCREMENT N2
$WROUT WRITE DESCRIPTOR ACCEPTED
ALREADY IN DOC ?
YES
STORE EXCLUSION BITS FROM PRE FIX - OR CODED VALUE TO STORAGE
P1 1
NO
STORE IT INCREMENTING M
P1 1

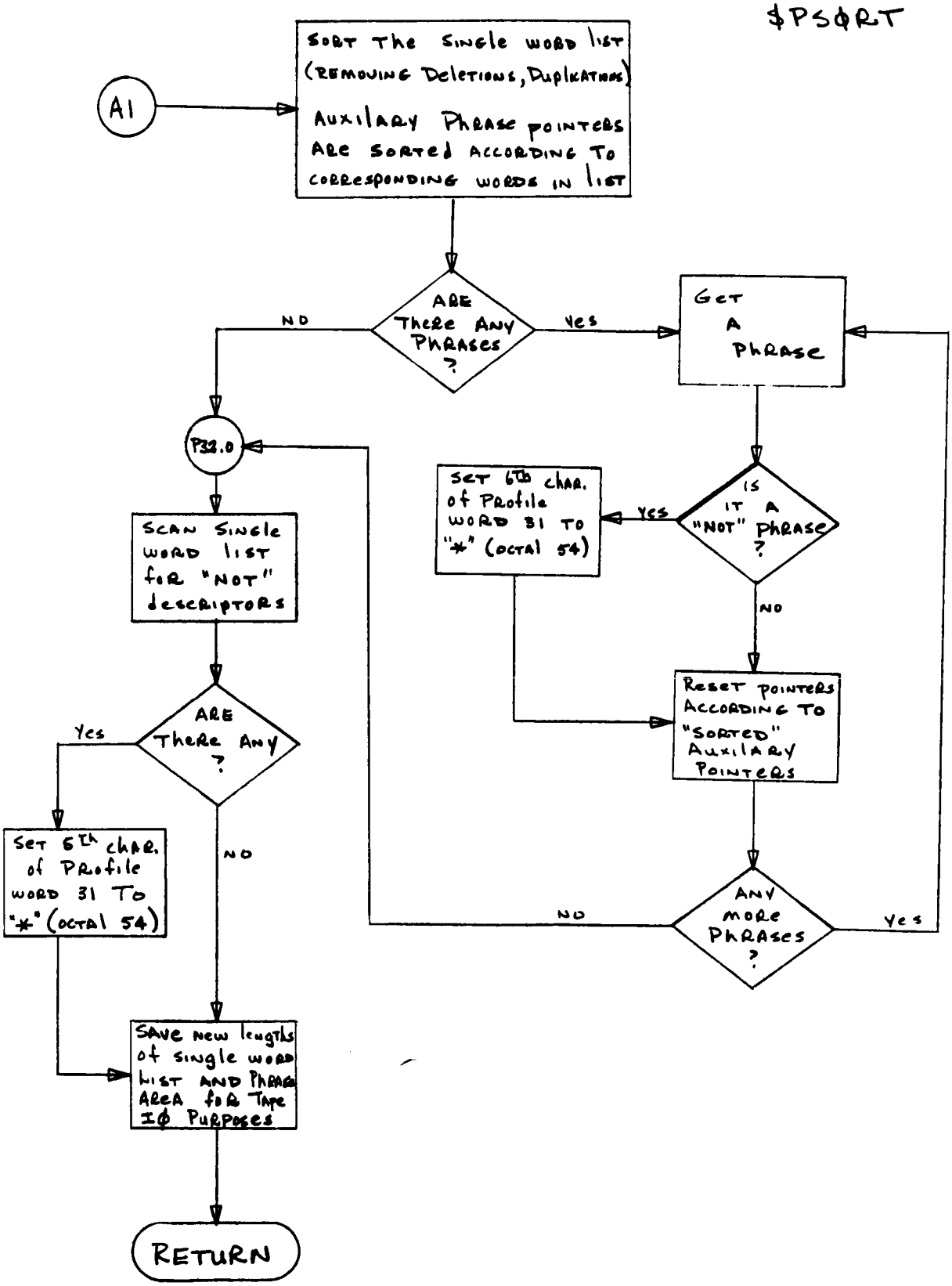

P2 1
MOVE M.P. + SUB PHRASES TO WORKING AREA
STORE WORD COUNT IN CR1, xR3
PRC
GET SUBPHRASE
BIT 33 ON ?
NO
YES
INCREMENT CR2
FINISHED SCAN ?
NO
YES
CR2/CR1 >70%
NO
$WROUT
WRITE DESCRIPTOR REJECTED
P1 1
YES
MAIN PHRASE ALREADY IN DOC ?
YES
P1 1
NO
INCREMENT N2
1
1
STORE MAIN PHRASE
GET SUBPHRASE
ALREADY IN DOCUMENT ?
YES
OR A 2 TO STORAGE
NO
STORE SUB-PHRASE WITH A 2 IN 32-35 INCREMENT M
STORE POINTER TO SUBPHRASE
FINISHED (X3)
NO
YES
$WROUT
WRITE DESCRIPTOR ACCEPTED
P1 1


P 2
2
SPC
ONE WORD ?
NO
SENSE LIGHT ONE ON
YES
ALREADY AMONG WORDS ?
1
NO
SENSE LIGHT ON ?
YES
MULTI-WORD PUT A 6 IN BITS 33-35
NO
ONE-WORD PUT A 5 IN BITS 33-35
STORE IT AND INCREMENT M
P 1
3
1
SENSE LIGHT ON ?
YES
MULTI-WORD PUT A 6 IN BITS 33-35
NO
ONE WORD PUT A 5 IN BITS 33-35
STORE PREFIX (EXCLUSION BITS)
OR CODED VALUE TO STORAGE
P 1
3

Subroutine PSORT

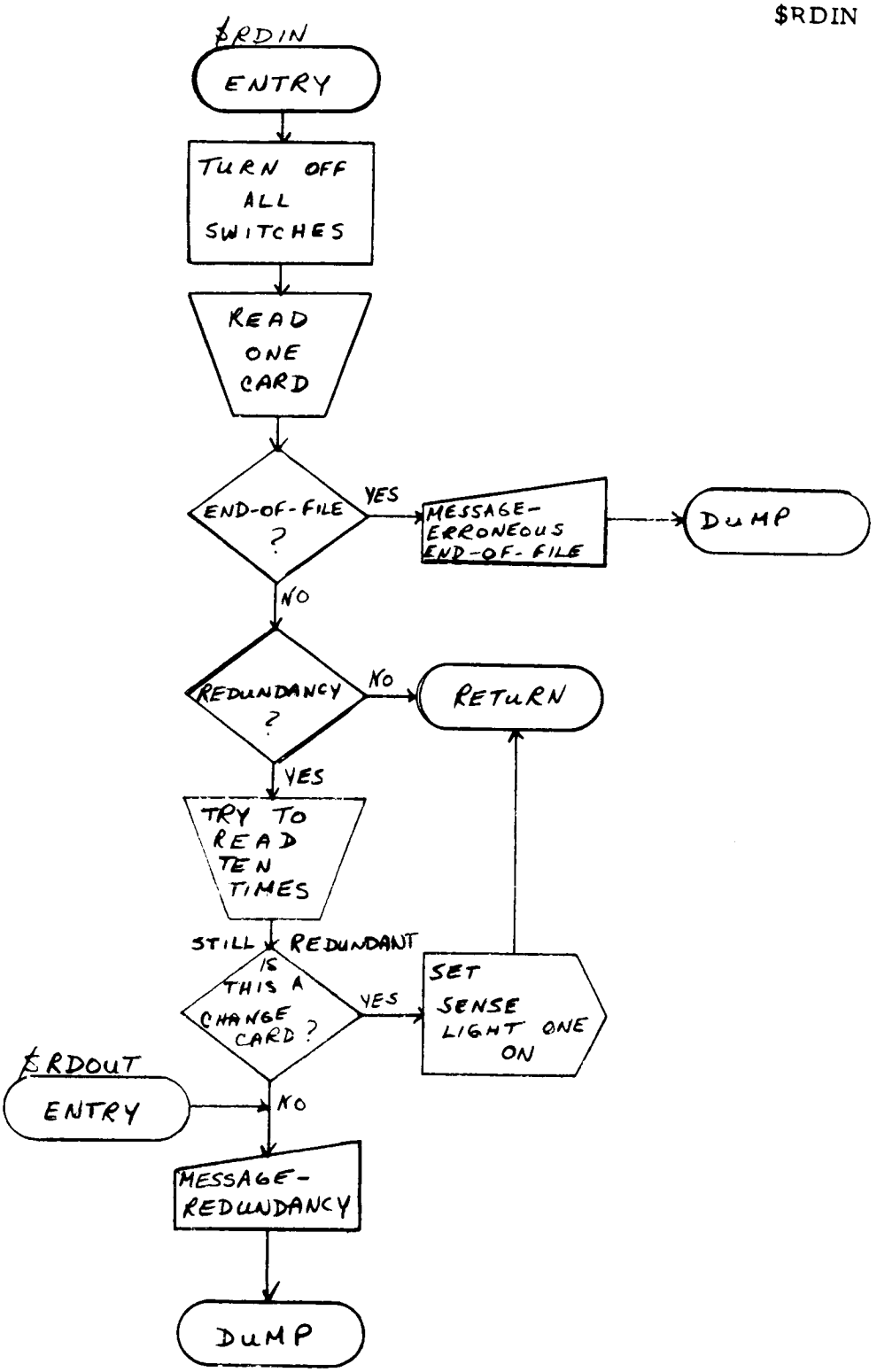
PSORT checks coded profiles (document or user) submitted to it to ascertain that the single-word list is sorted and that phrase pointers are set accordingly. If not, PSORT sorts and sets them.


ENTRY
PS$RT
Get L, M, and N from Profile word 27
M+N >1 ?
NO
RETURN
YES
START
Set internal parameters (profile location, size, etc.)
Set Profile word 31 to zero in 4th, 5th, 6th characters
ARE There any Phrases
yes (skip≠0)
Generate Auxilary Phrase Pointers
Remove Phrase indicator (bit 34=0) throughout single word list
Get A Phrase
NO (skip=0)
for each single word (in list) Referenced by Phrase Pointers, Set bit 34 = 1
NO
has it been deleted ?
Yes
ANY more Phrases ?
NO
Yes
A1


A1
SORT THE SINGLE WORD LIST (REMOVING DELETIONS, DUPLICATIONS) AUXILARY PHRASE POINTERS ARE SORTED ACCORDING TO CORRESPONDING WORDS IN LIST
ARE THERE ANY PHRASES ?
NO
YES
GET A PHRASE
P32.0
SCAN SINGLE WORD LIST FOR "NOT" DESCRIPTORS
IS IT A "NOT" PHRASE ?
YES
SET 6TH CHAR. of PROFILE WORD 31 TO "*" (OCTAL 54)
NO
RESET POINTERS ACCORDING TO "SORTED" AUXILARY POINTERS
ARE THERE ANY ?
YES
SET 5TH CHAR. of PROFILE WORD 31 TO "*" (OCTAL 54)
NO
ANY MORE PHRASES ?
NO
YES
SAVE NEW LENGTHS of SINGLE WORD LIST AND PHRASE AREA FOR TAPE I/O PURPOSES
RETURN

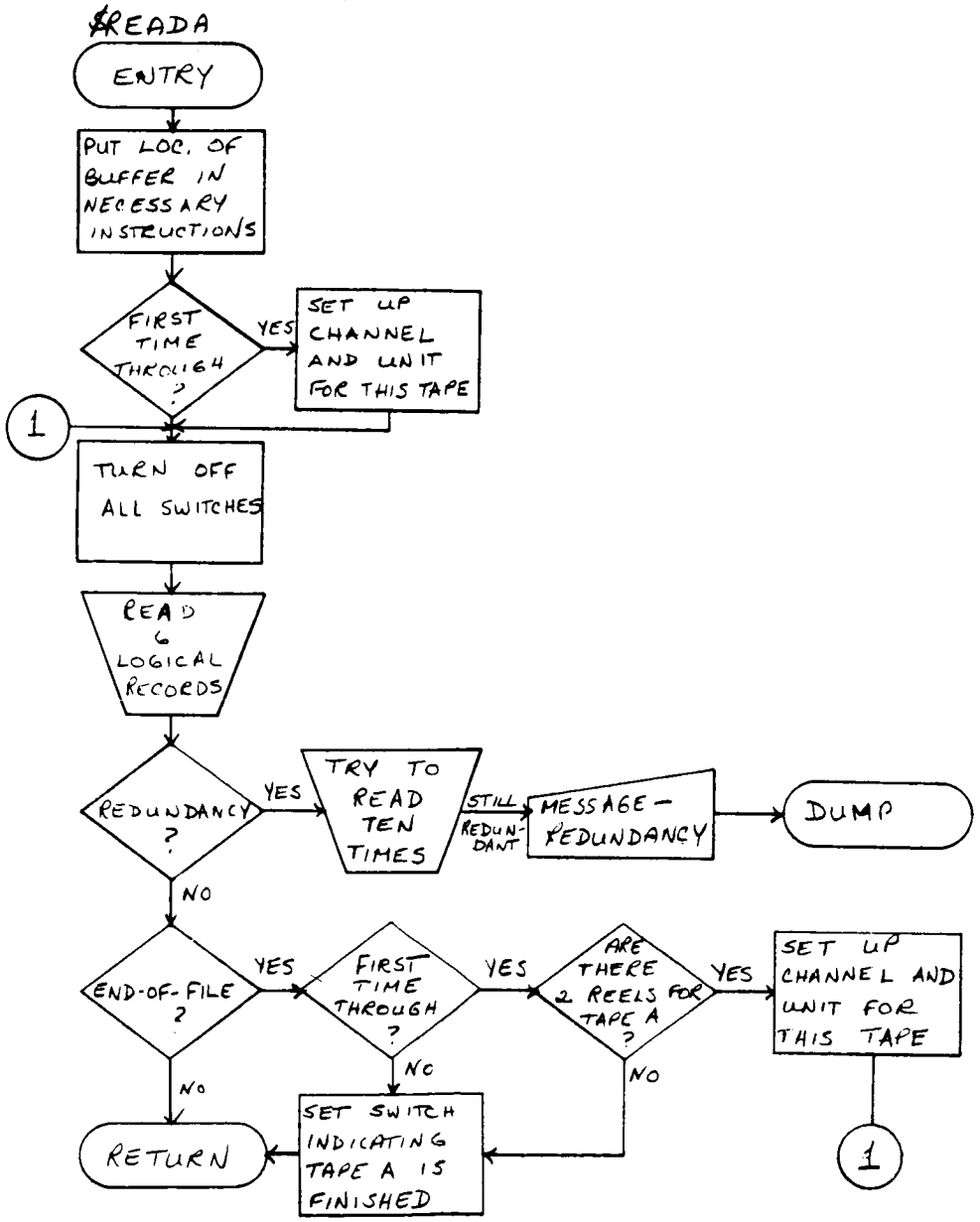
Subroutine RDIN

This subroutine, used by MAIN and GTUSE, reads the system input tape. On any redundancy, an attempt is made to read the record ten times. If the redundant condition persists for a control card, a message indicating the redundancy is printed and a core dump is taken. RDIN also checks for EOF and prints a message if the condition exists and dumps core.


$RDIN
ENTRY
TURN OFF ALL SWITCHES
READ ONE CARD
END-OF-FILE ?
YES
MESSAGE-ERRONEOUS END-OF-FILE
DUMP
NO
REDUNDANCY ?
NO
RETURN
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
IS THIS A CHANGE CARD ?
YES
SET SENSE LIGHT ONE ON
$RDOUT
ENTRY
NO
MESSAGE-REDUNDANCY
DUMP

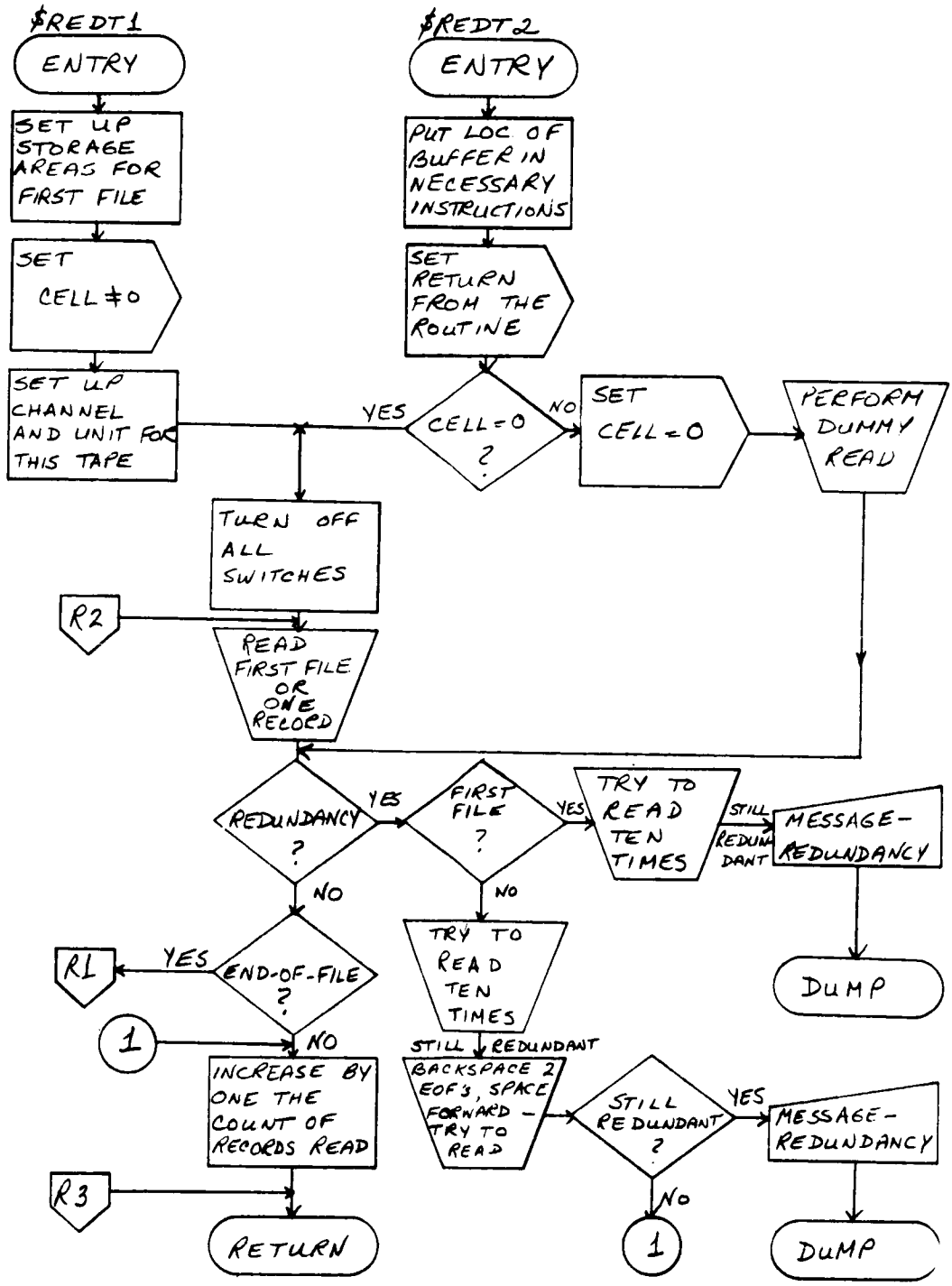
Subroutine READA

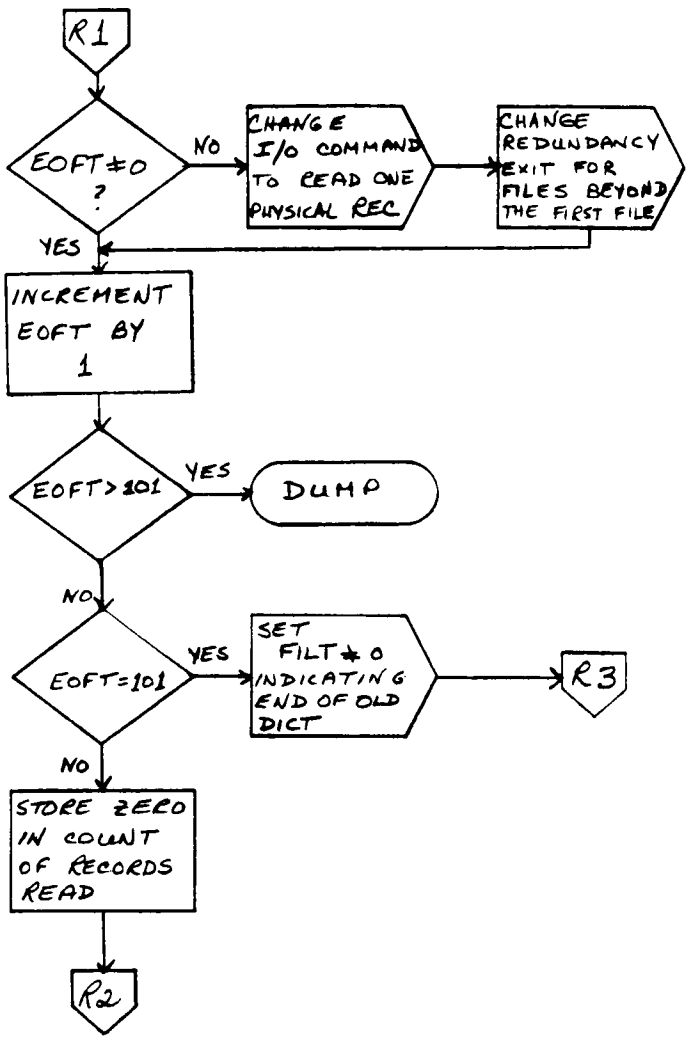
This subroutine reads a physical record consisting of six logical records from tape A (sorted input tape). On a redundancy condition, an attempt is made to read a record ten times before a message is given indicating the redundancy and a dump of core is taken. The routine allows tape A to have two reels.


$READA
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
1
TURN OFF ALL SWITCHES
READ 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
MESSAGE - REDUNDANCY
DUMP
NO
END-OF-FILE ?
YES
FIRST TIME THROUGH ?
YES
ARE THERE 2 REELS FOR TAPE A ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
1
NO
NO
NO
RETURN
SET SWITCH INDICATING TAPE A IS FINISHED

Subroutine REDT

This subroutine reads the old vocabulary. It has two entry points, REDT1 to read the first file and REDT2 to read the other one hundred files. A redundancy condition on the first file causes an attempt to be made to read the file ten times. If this fails, a message is printed and a dump taken. A redundant condition on any other file causes the tape to be reread ten times if necessary. If this fails, the tape is backspaced over two EOF's, spaced forward again, and another attempt is made to read. If the redundancy still exists a message is given and a dump taken.


$REDT1
ENTRY
SET UP STORAGE AREAS FOR FIRST FILE
SET CELL ≠ 0
SET UP CHANNEL AND UNIT FOR THIS TAPE
$REDT2
ENTRY
PUT LOC OF BUFFER IN NECESSARY INSTRUCTIONS
SET RETURN FROM THE ROUTINE
CELL = 0 ?
YES
NO
SET CELL = 0
PERFORM DUMMY READ
TURN OFF ALL SWITCHES
R2
READ FIRST FILE OR ONE RECORD
REDUNDANCY ?
YES
FIRST FILE ?
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
MESSAGE-REDUNDANCY
DUMP
NO
NO
TRY TO READ TEN TIMES
STILL REDUNDANT
BACKSPACE 2 EOF's, SPACE FORWARD - TRY TO READ
STILL REDUNDANT ?
YES
MESSAGE-REDUNDANCY
DUMP
No
1
RL
YES
END-OF-FILE ?
1
NO
INCREASE BY ONE THE COUNT OF RECORDS READ
R3
RETURN


R1
EOFT≠0 ?
NO
CHANGE I/O COMMAND TO READ ONE PHYSICAL REC
CHANGE REDUNDANCY EXIT FOR FILES BEYOND THE FIRST FILE
YES
INCREMENT EOFT BY 1
EOFT>101
YES
DUMP
NO
EOFT=101
YES
SET FILT ≠ 0 INDICATING END OF OLD DICT
R3
NO
STORE ZERO IN COUNT OF RECORDS READ
R2

Subroutine SORT

General Specifications

The SORT subroutine is a generalized internal sort and tape merge program that sorts fixed-length blocked or unblocked binary records. The SORT routine is entered in Phases 3, 5, and 6 of the program. In Phase 3, it sorts user and document profiles and vocabulary changes into coded descriptor order. In Phases 5 and 6, it sorts edited document profiles and edited user profiles, respectively, into profile and serial number order.

The sort key must be located at the beginning of each logical record and must be a multiple of six bits in length. Input and output files may each use one alternate unit. All input tapes must have a BCD trailer label. For multi-reel files, each tape must end with an end-of-file and a 120-character 1EOR record. The last reel, or the single reel of a one-reel file should have an end-of-file and a 120-character 1EOT record. For the merge pass, the package requires four intermediate tapes, two on channel A and two on channel B, the blocking of these tapes may be different from the initial or final blocking.

The SORT package consists of three major subroutines, EDIT, SORT, and WRITE. It requires the FORTRAN II Input/Output package IOP for its I/O functions, and the subroutine DMP for catastrophic errors. The SORT package must be assembled by the user, who provides variables in cards 60-400 as required (see listing). It uses approximately 1200)10 locations plus BSS areas, the size of which depends on the physical size of the records to be sorted. All core not needed by the object program should be allocated to SORT for internal sorting. The calling sequence to SORT is TSX SORT,4 with a return of 1,4.

Subroutine EDIT

This subroutine has two entries, EDITA and EDIT, which are the first and second executable instructions. EDITA is the entry for the first call, to accomplish any initialization necessary in view of the fact that the sort may be restarted due to tape error. The routine places in the AC decrement (bits 0-18) the number of words in the record to be sorted, and in the MQ address (19-36) the number of characters in the sort key. It then returns to 1,4. EDIT is the normal entry. The calling sequence is TSX EDIT,4, end-of-file return, normal return.

Each time it is called, the subroutine will store a record starting at 1,2. It reads as much or as little data from the mediary tape as it needs to. The record placed at 1,2 is arranged with the sort key at the beginning, since the sort program will sort on only one field. Conversions to modify the natural sequence or complementing of fields to achieve a descending sort may be done prior to storing the record at 1,2. If there is no more data, return is to 1,4. The normal return is to 2,4. The subroutine also handles any errors encountered as described below.

Subroutine SORT

A replacement chain sort is used for the internal (string-forming) phase of the sort program. The chain is formed as follows: With each record of data is associated one word of control information. The decrement of this word contains the location of the next smaller record currently in core. The address portion contains the location of the next greater record (actually the two's complement of the address of the word preceding the first one of the record in question). The decrement of the control word of the smallest record in core (the beginning of the chain) contains zero, as does the address of the control word of the largest record (the end of the chain). Thus, given the location of the starting record of the chain, one can reference each record in the desired logical sequence, and the data is effectively sorted. Note that the physical arrangement of the data in memory is irrelevant to the structure of the chain. Note also that the insertion of a new record into the chain requires the creation of a control word for the new record and the modification of the control words of the two records which will immediately precede and follow it in the chain. No physical movement or shifting of the data is necessary.

The sort program starts by storing records sequentially in memory, preceding each one by its control word. Each record received from EDIT is immediately put in the chain by the subroutine INSERT. Since some pre-sequencing is expected to exist in the data, the INSERT subroutine starts by comparing the new record with the one most recently inserted. Depending on the result of this comparison, the subroutine searches the chain in either the forward (increasing) or backward (decreasing) direction until the proper place for the new record is found. In the case of equal records, the order of input is preserved. The new record is then inserted in the chain, as described above. A record is kept of the location of the starting member of the chain, and updated when necessary.

If all the input has been read in before available core space is filled, the replacement phase and tape merge, described below, are skipped, and the routine goes directly into the WRITE subroutine. Otherwise, replacement and tape merge are necessary.

The replacement phase of the internal sort is started when there is room for only one more record in memory. Starting with the first merge tape and the first record in the chain, the procedure is as follows:

(1) Write the first record in the chain on the merge tape;
(2) Read a new (replacement) record into the one vacant space in core;
(3) Insert the new record in the chain (if it is greater than or equal to the previously written record, it will be in the same string. If less, it will be in the next string);
(4) Remove the previously written record from the chain, vacating its space;
(5) Repeat from step (1), writing out the follower in the chain of the previously written record.

This procedure continues until either the end of the input is reached or the last member of the chain has been written out. In the latter case, the chain is now composed of those replacement records which were less than the record just written when they were read in (otherwise, they would have been written out in that string and themselves replaced). An end-of-file is written after the completed string, and a new string is started on the next merge tape, again beginning with the first member of the chain. When there is no more input, the data is written as above without being replaced. The current string is finished and, unless the input end-of-file had coincided with the start of a new string, one final string is written.

Thus, throughout most of the internal sort phase, the operations of reading, writing and sorting are overlapped and all take place in the same storage area. The lengths of the strings produced depend on the pre-sequencing of the input. In the worst case (perfect inverse order) the strings contain as many records as can be held in memory at one time. In any other case, the strings will be considerably longer.

The technique used is that of a simple balanced merge. The final pass, in which some number of strings $\leq$ the merge order (one string to a tape) are merged into one output string, gives the sorted data directly to the WRITE subroutine.

Subroutine WRITE

This subroutine has one entry, WRITE, which is the first executable instruction. Each time it is called, it finds one sorted record of data starting at 1,2, in ascending logical order. The routine may re-convert or complement the sort key, as necessary, and then will write the appropriate record(s) of the output report on the system's output tape. When all data has been processed, the sort program must make one last entry to WRITE with a word of all one's, $777777777777)_8$, at 1,2.

The FORTRAN II Input/Output Package (IOP) is called into core by using the * IOP card in the deck setup. The program uses IOP to take care of much of its input/output functions.

Error Messages

The sort program keeps two checks on its processing. First, the records input are counted, and this number compared with the number processed by each merge pass and the number eventually output. This is a check for records lost or duplicated due to program or tape error. The program also checks that records written out, either on merge tapes or given to WRITE are, in fact, in the proper sequence. Violations of the sort order could be due to program error or undetected tape error.

UNEXPECTED EOF ON MEDIARY TAPE.
The EDIT subroutine took the EOF return the first time it was entered. There may have been no data at all on input tape or none valid.

ERROR READING MERGE TAPE XX
This message is written after an error return from a BBREAD of a merge (scratch) tape. If it is not the last pass of the sort and it is the first time this has occurred, the program prints on-line: ERROR READING TAPE XX. MOUNT NEW TAPE AND PRESS START TO CONTINUE JOB. The program then stops at HPR 0. The operators are to change tapes and let the program proceed, in which case it prints "RESTARTING SORT" off-line and re-trys the sort from the beginning. If, however, this re-occurs or occurs the first time during the last pass, the program will follow the first message with "END PROCESSING" and will transfer $DMP.

UNEXPECTED EOF ON MERGE TAPE XX.
END PROCESSING.
This unlikely message indicates that the sort program's count of the number of strings on the merge tape was incorrect. This could be due to program error, machine error or tape error. The program transfers to $DMP.

PH.X RECORD COUNTS DISAGREE. / END PROCESSING.
PH.X DATA OUT OF SORT. / END PROCESSING.
Phase 1 (PH.X) is the internal sort and string-forming section (the first pass over the data); Phase 2 is the tape merge. Phase 3 is the final merge pass and writing of the report itself. The errors are most likely due to tape failure, although they could be caused by improper tampering with data and storage by the EDIT and WRITE subroutines. The program transfers to $DMP.

Sort Tape Formats

Tape A - All Input Data

Tape A is sorted only on word 1, that is, according to the coded value of the descriptor. Hence all header records will appear before descriptor records on the sorted Tape A.

Descriptor Records:

Word	
1	Coded value of descriptor
2	Same as the third word put out by CODER. (Bits S1-17 have type, col. 71 of input card, etc. Bits 18-35 have serial number.)
3	Zero if dictionary change, otherwise the document or user profile number.
4-23	Twenty word alphanumeric descriptor. (If this descriptor is a subphrase, these twenty words will be either blanks or zeros depending on the output from CODER.)

24	Coded value if this is a dictionary secondary descriptor, otherwise blanks or zeros.
25-27	All blanks or zeros.

Header Records:

Word	
1	All zeros
2	Same as for a descriptor record
3	Document or user profile number
4-27	Twenty-four word heading.

Tapes B and C

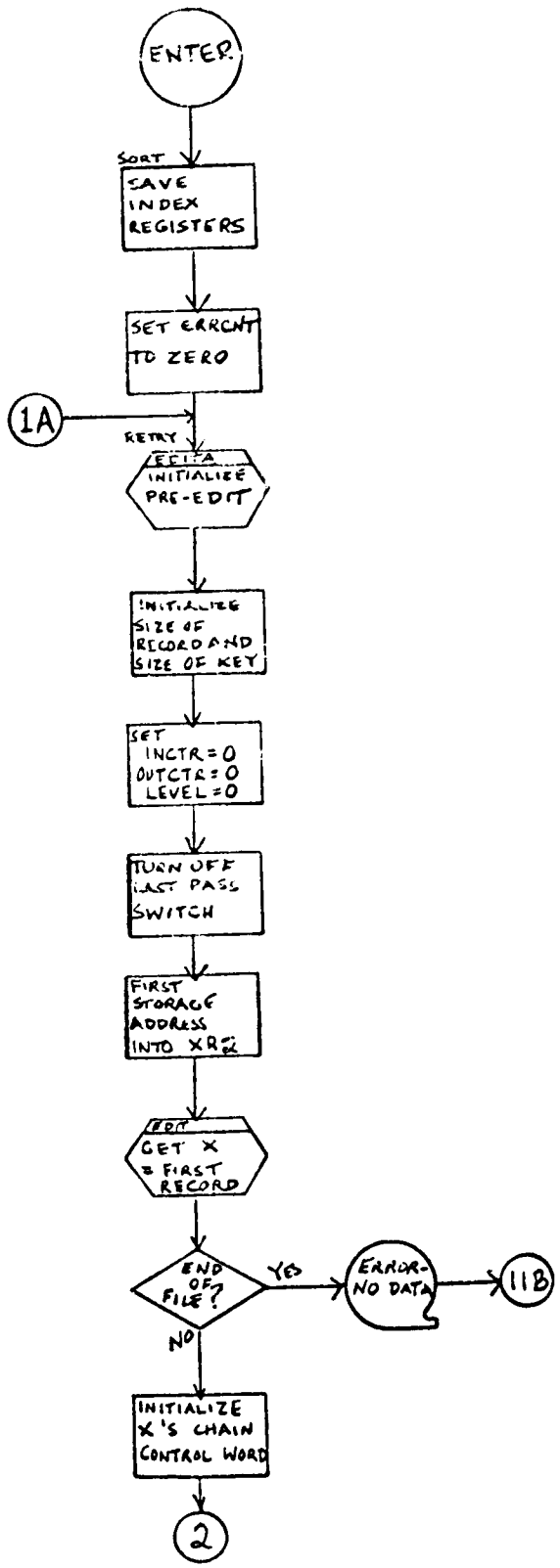
Tape B has document records and Tape C has user records. Both documents and users have the same basic format. Tapes B and C are sorted on profile number and serial number. Hence all descriptors having the same profile number will be together preceded by the header for that profile.

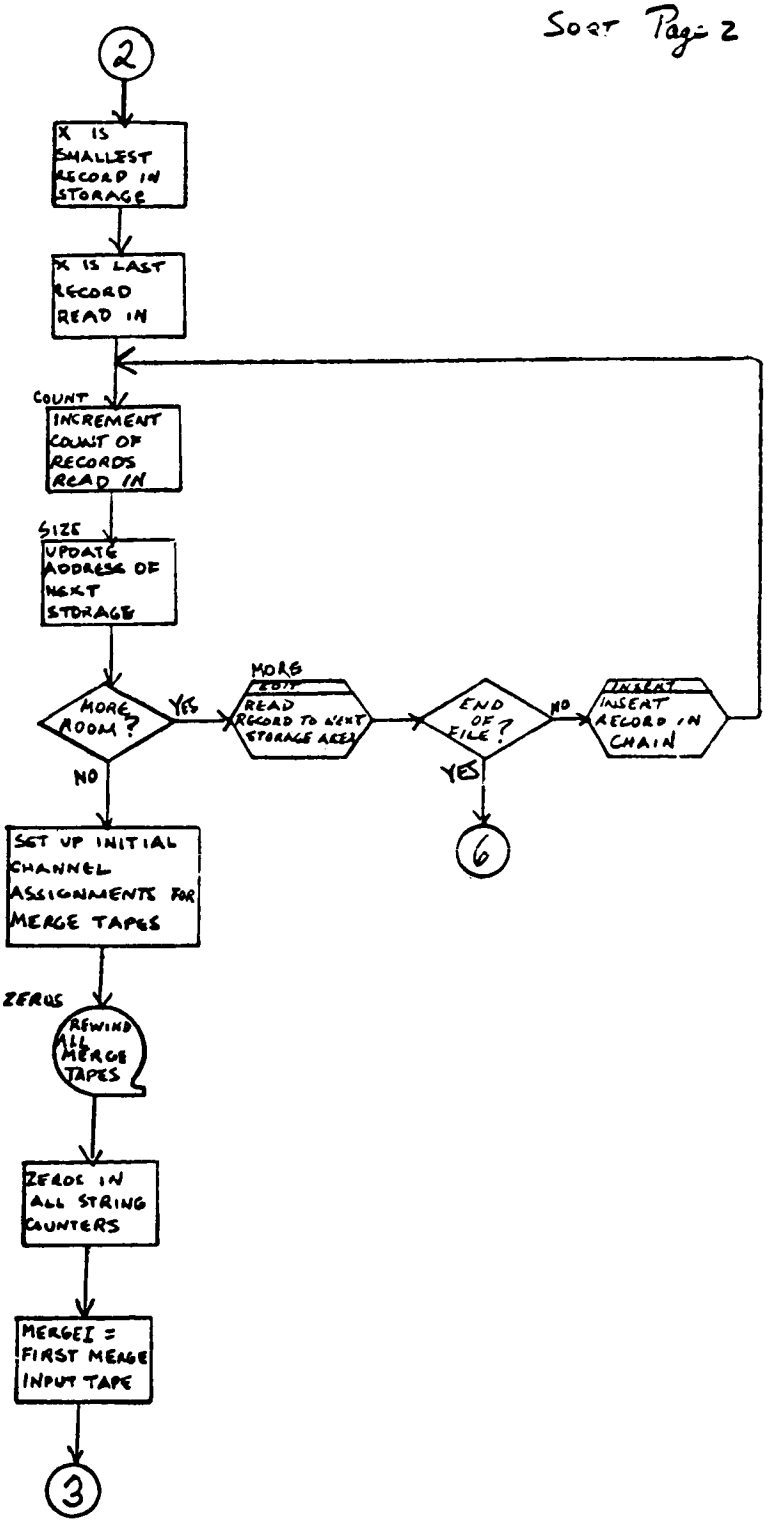
Descriptor Records:

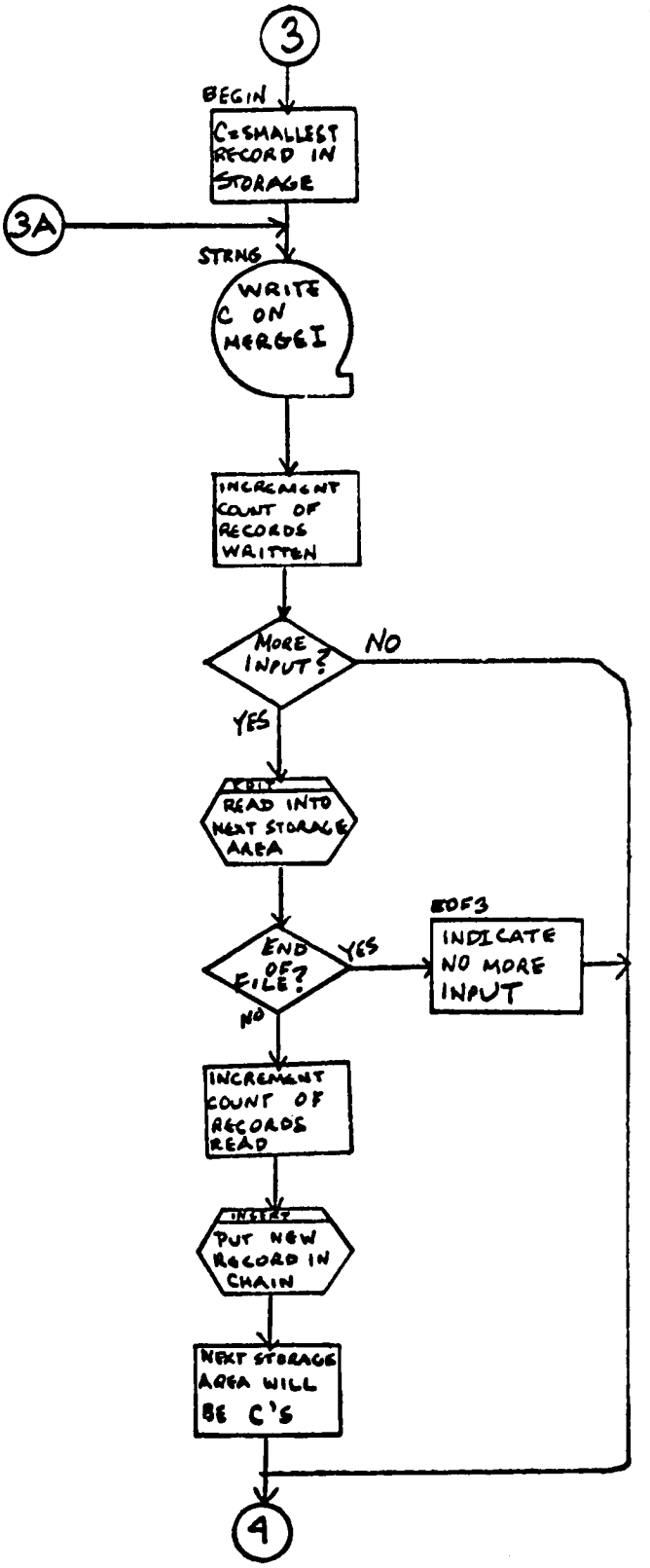
Word	
1	Document or user profile number
2	Has the same content as word 3 of the CODER output but with the two halves of the word interchanged: bits S1-17, serial number, bits 18-35, various bits indicating type, col. 71, etc.
3	Coded value of descriptor
4-23	Twenty-word alphanumeric descriptor (or all blanks if a subphrase.)
24-27	All blanks.

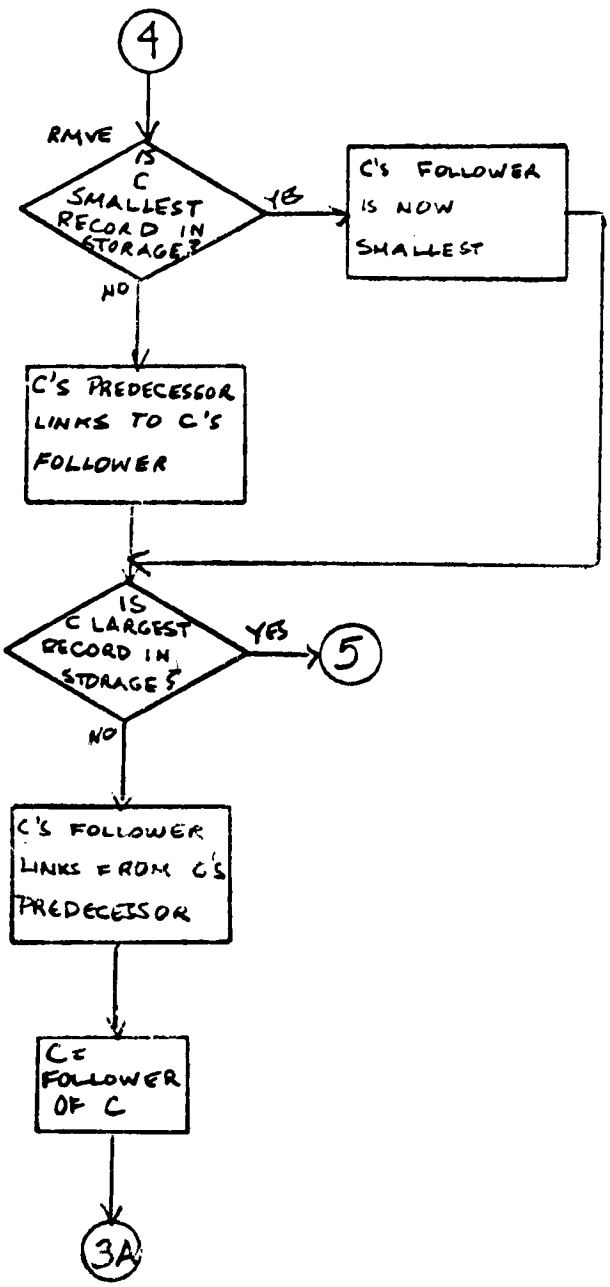
Header Records:

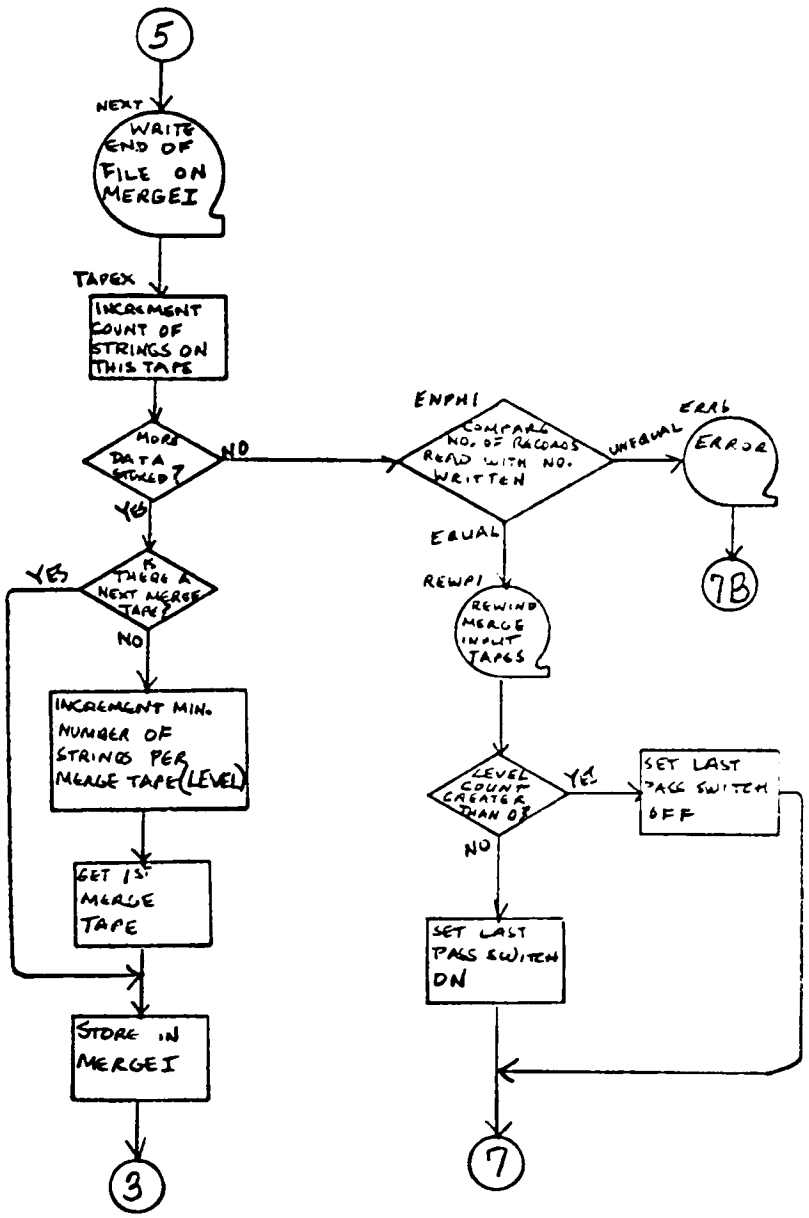
Word	
1-2	Same as descriptor records
3	All zeros
4-27	Twenty-four word heading.

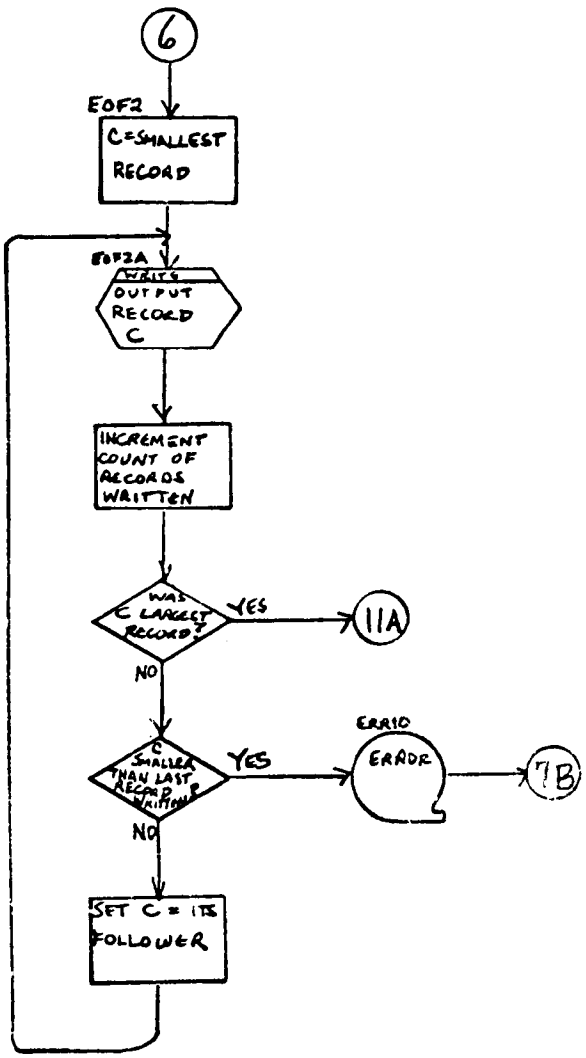

ENTER
SORT
SAVE INDEX REGISTERS
SET ERRCNT TO ZERO
1A
RETRY
EDIT A
INITIALIZE PRE-EDIT
INITIALIZE SIZE OF RECORD AND SIZE OF KEY
SET INCTR=0 OUTCTR=0 LEVEL=0
TURN OFF LAST PASS SWITCH
FIRST STORAGE ADDRESS INTO XR 2
EDIT
GET X = FIRST RECORD
END OF FILE?
YES
ERROR-NO DATA
11B
NO
INITIALIZE X'S CHAIN CONTROL WORD
2

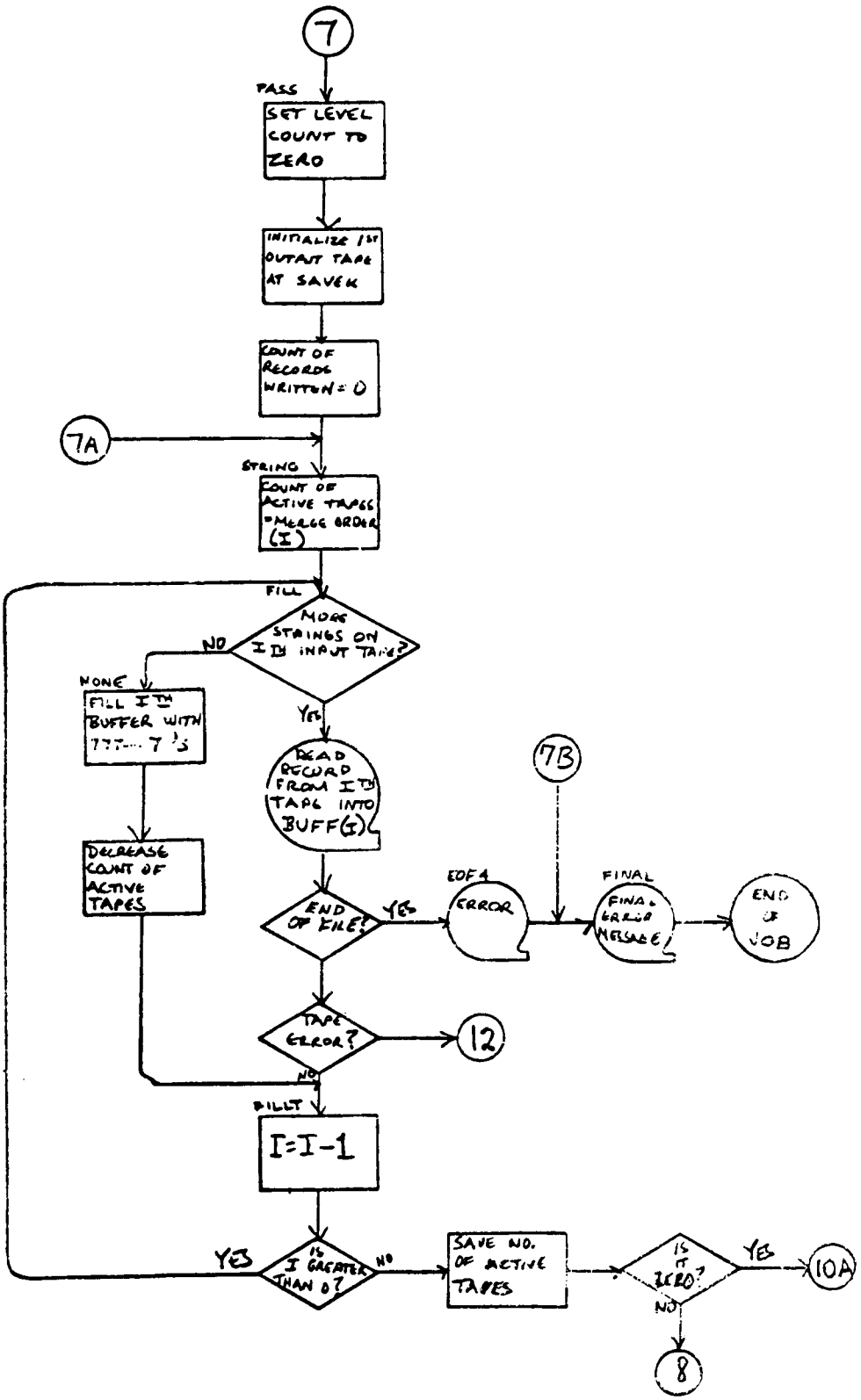

2
X IS SMALLEST RECORD IN STORAGE
X IS LAST RECORD READ IN
COUNT
INCREMENT COUNT OF RECORDS READ IN
SIZE
UPDATE ADDRESS OF NEXT STORAGE
MORE ROOM?
YES
MORE EDIT
READ RECORD TO NEXT STORAGE AREA
END OF FILE?
NO
INSERT
INSERT RECORD IN CHAIN
YES
6
NO
SET UP INITIAL CHANNEL ASSIGNMENTS FOR MERGE TAPES
ZEROS
REWIND ALL MERGE TAPES
ZEROS IN ALL STRING COUNTERS
MERGEI = FIRST MERGE INPUT TAPE
3

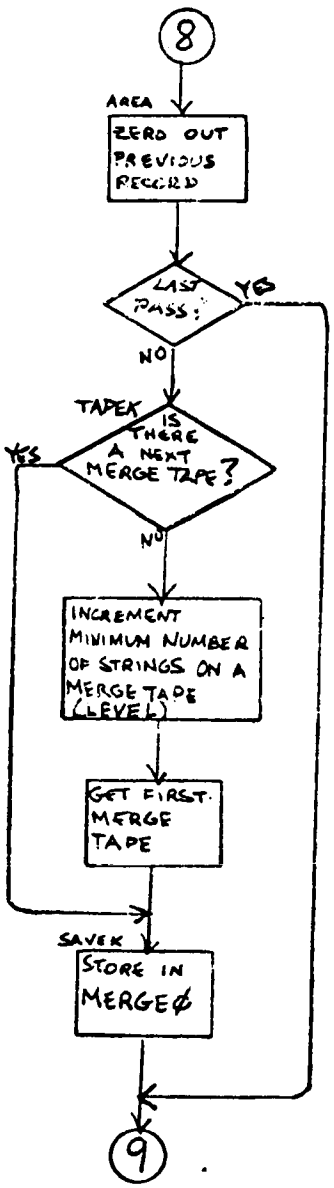

3
BEGIN
C=SMALLEST RECORD IN STORAGE
3A
STRNG
WRITE C ON MERGE1
INCREMENT COUNT OF RECORDS WRITTEN
MORE INPUT?
NO
YES
READ INTO NEXT STORAGE AREA
END OF FILE?
YES
EOF3
INDICATE NO MORE INPUT
NO
INCREMENT COUNT OF RECORDS READ
PUT NEW RECORD IN CHAIN
NEXT STORAGE AREA WILL BE C'S
4

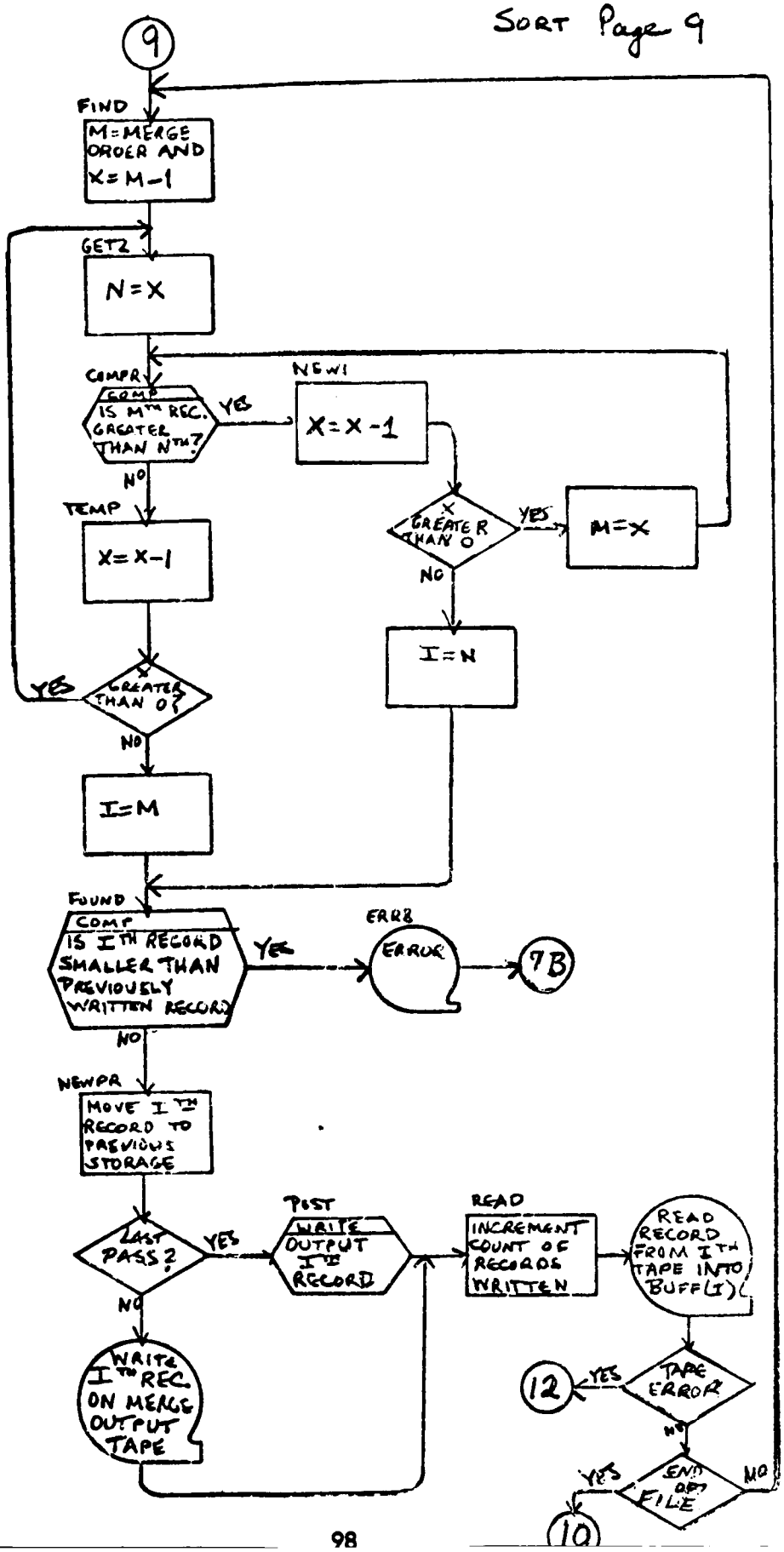

4
RMVE
IS
C
SMALLEST
RECORD IN
STORAGE?
YES
C's FOLLOWER
IS NOW
SMALLEST
NO
C's PREDECESSOR
LINKS TO C's
FOLLOWER
IS
C LARGEST
RECORD IN
STORAGE?
YES
5
NO
C's FOLLOWER
LINKS FROM C's
PREDECESSOR
C=
FOLLOWER
OF C
3A

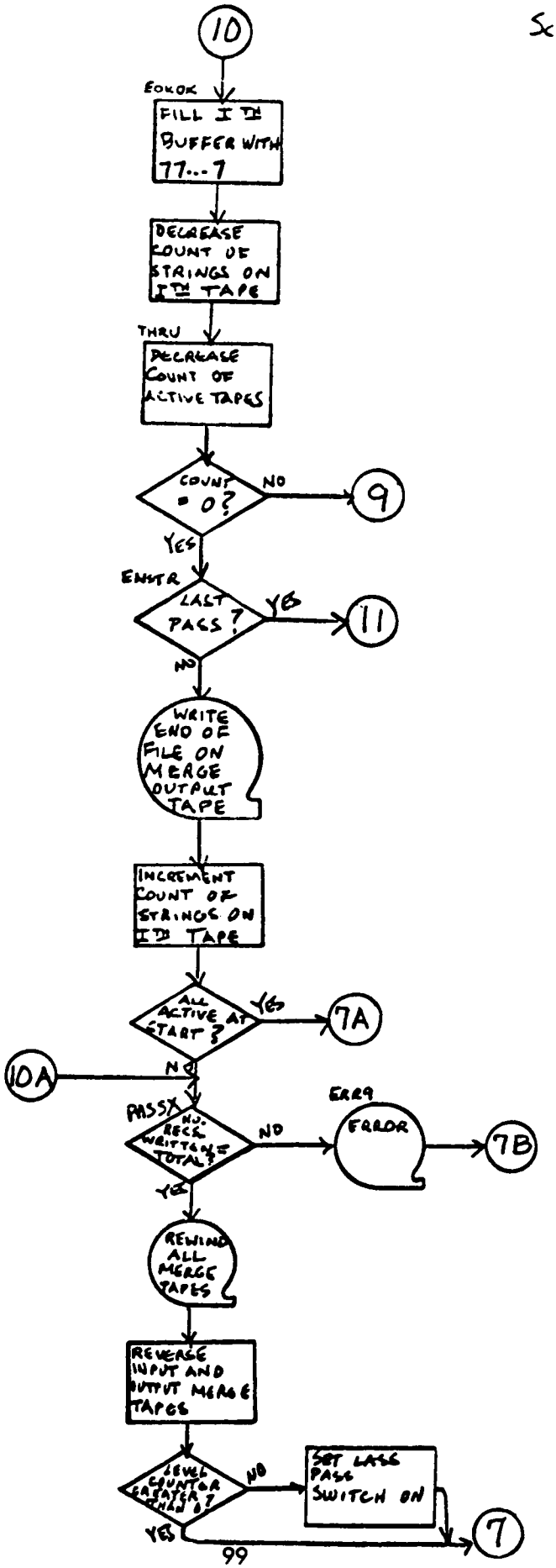

5
NEXT
WRITE END OF FILE ON MERGEI
TAPEX
INCREMENT COUNT OF STRINGS ON THIS TAPE
MORE DATA STORED?
NO
YES
ENPH1
COMPARE NO. OF RECORDS READ WITH NO. WRITTEN
UNEQUAL
ERA6
ERROR
7B
EQUAL
REWP1
REWIND MERGE INPUT TAPES
IS THERE A NEXT MERGE TAPE?
YES
NO
INCREMENT MIN. NUMBER OF STRINGS PER MERGE TAPE (LEVEL)
GET 1ST MERGE TAPE
STORE IN MERGEI
3
LEVEL COUNT GREATER THAN 0?
YES
NO
SET LAST PASS SWITCH OFF
SET LAST PASS SWITCH ON
7

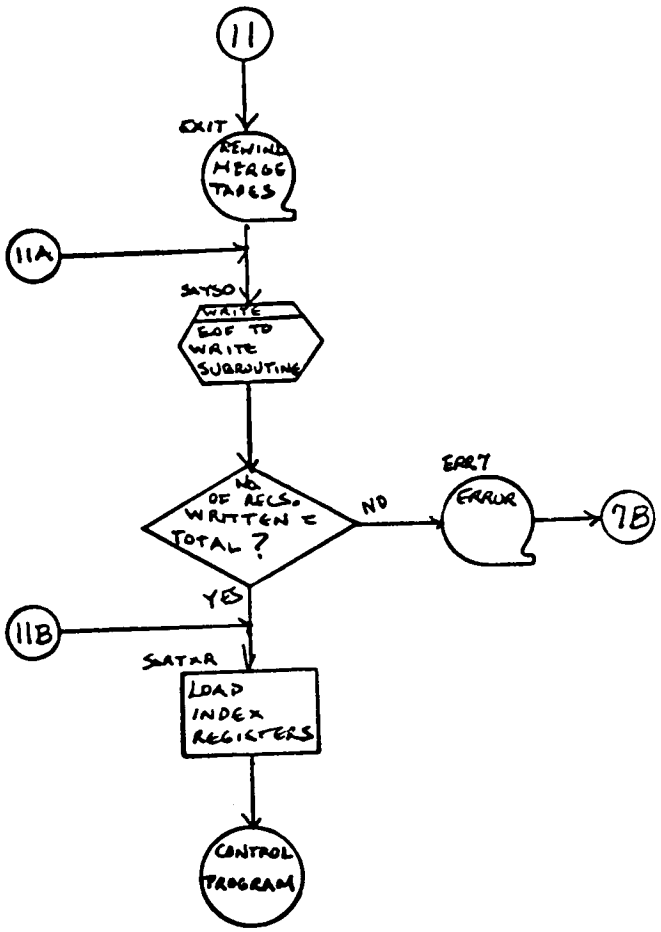

6
EOF2
C=SMALLEST RECORD
EOF2A
WRITE OUTPUT RECORD C
INCREMENT COUNT OF RECORDS WRITTEN
WAS C LARGEST RECORD?
YES
11A
NO
C SMALLER THAN LAST RECORD WRITTEN?
ERR10
YES
ERROR
7B
NO
SET C = ITS FOLLOWER

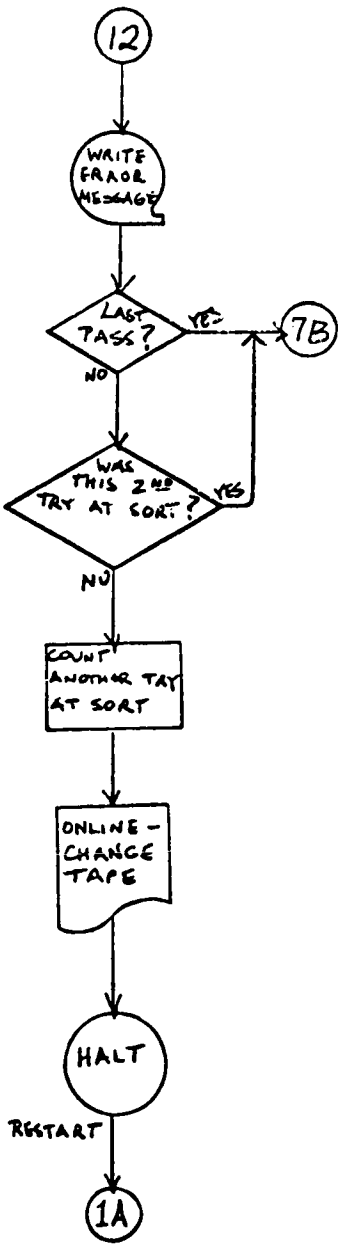

7
PASS
SET LEVEL COUNT TO ZERO
INITIALIZE 1ST OUTPUT TAPE AT SAVE K
COUNT OF RECORDS WRITTEN = 0
7A
STRING
COUNT OF ACTIVE TAPES = MERGE ORDER (I)
FILL
MORE STRINGS ON I TH INPUT TAPE?
NO
NONE
FILL I TH BUFFER WITH 777---7's
DECREASE COUNT OF ACTIVE TAPES
YES
READ RECORD FROM I TH TAPE INTO BUFF(I)
7B
END OF FILE?
YES
EOF 4 ERROR
FINAL
FINAL ERROR MESSAGE
END OF JOB
TAPE ERROR?
12
NO
FILLT
I=I-1
IS I GREATER THAN 0?
YES
NO
SAVE NO. OF ACTIVE TAPES
IS IT ZERO?
YES
10A
NO
8


8
AREA
ZERO OUT PREVIOUS RECORD
LAST PASS?
YES
NO
TAPEK
IS THERE A NEXT MERGE TAPE?
YES
NO
INCREMENT MINIMUM NUMBER OF STRINGS ON A MERGE TAPE (LEVEL)
GET FIRST MERGE TAPE
SAVEK
STORE IN MERGEØ
9

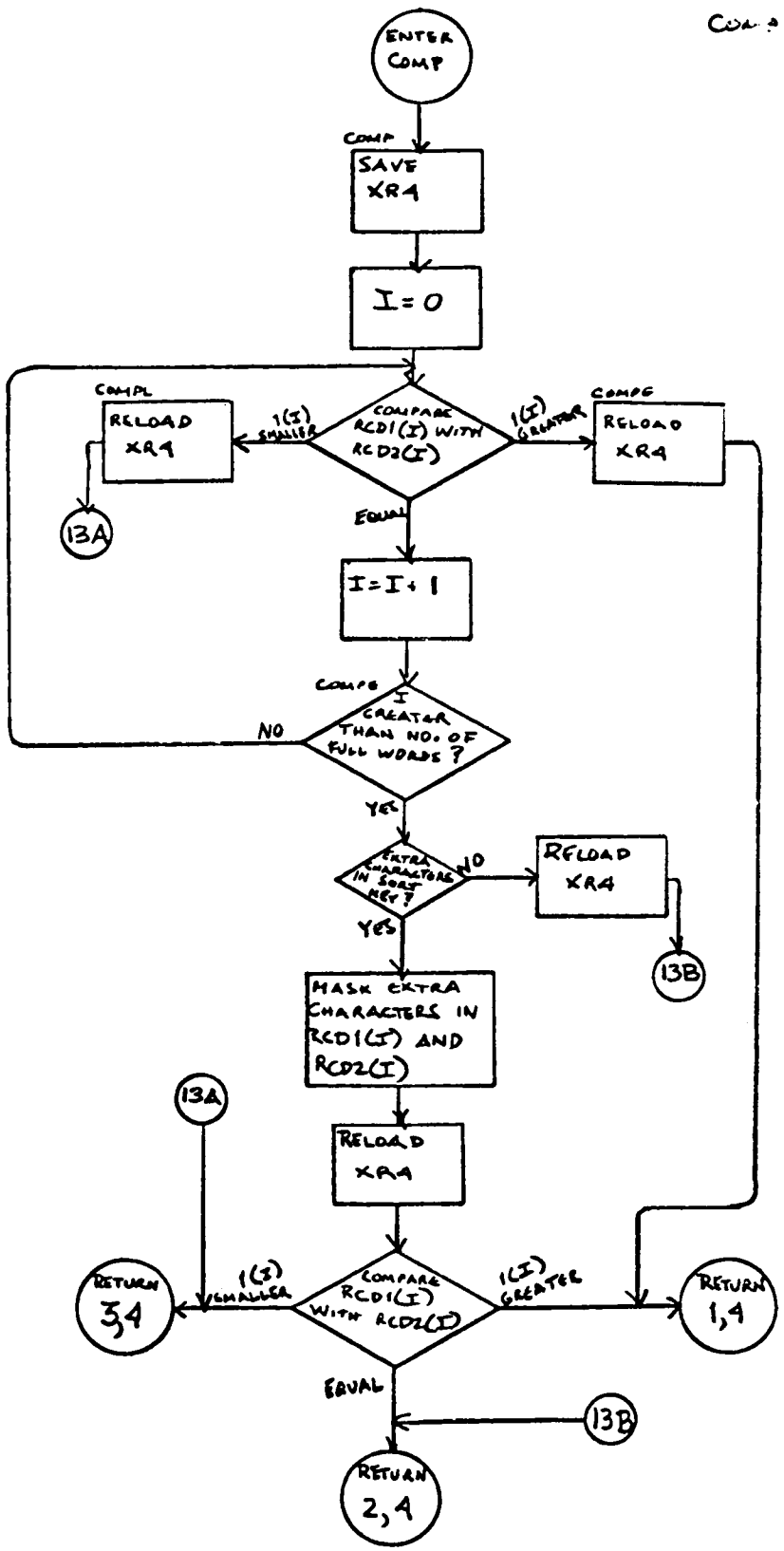

9
FIND
M=MERGE ORDER AND X=M-1
GET2
N=X
COMPR
COMP
IS Mth REC. GREATER THAN Nth?
YES
NEW1
X=X-1
X GREATER THAN 0
YES
M=X
NO
I=N
NO
TEMP
X=X-1
X GREATER THAN 0?
YES
NO
I=M
FOUND
COMP
IS Ith RECORD SMALLER THAN PREVIOUSLY WRITTEN RECORD
YES
ERR8
ERROR
7B
NO
NEWPR
MOVE Ith RECORD TO PREVIOUS STORAGE
LAST PASS?
YES
POST
WRITE OUTPUT Ith RECORD
READ
INCREMENT COUNT OF RECORDS WRITTEN
READ RECORD FROM Ith TAPE INTO BUFF(I)
NO
WRITE Ith REC. ON MERGE OUTPUT TAPE
12
YES
TAPE ERROR
NO
YES
END OF FILE
NO
10

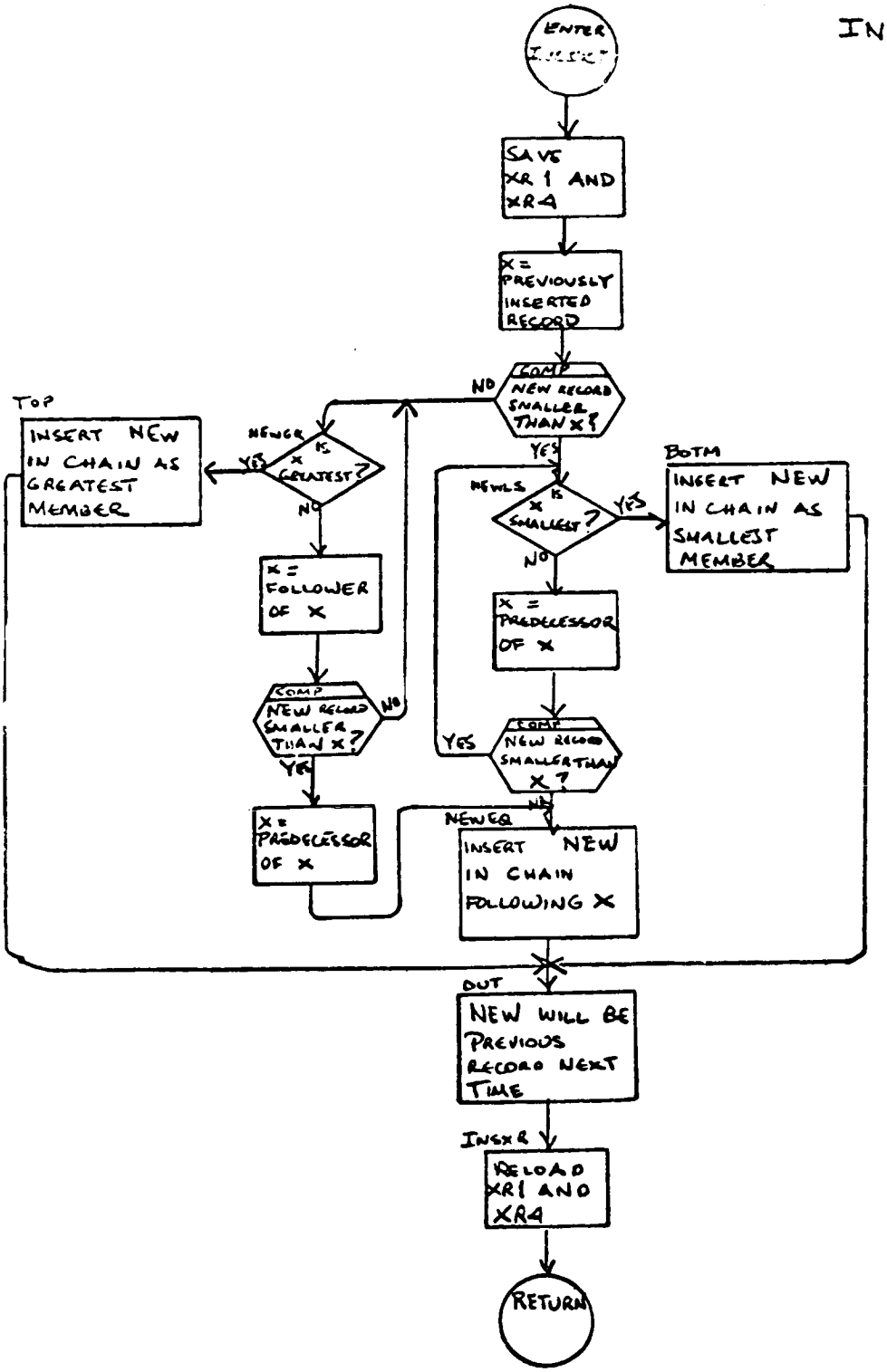

10
EOKOK
FILL I/O BUFFER WITH 77...7
DECREASE COUNT OF STRINGS ON I/O TAPE
THRU
DECREASE COUNT OF ACTIVE TAPES
COUNT = 0?
NO
9
YES
ENTR
LAST PASS?
YES
11
NO
WRITE END OF FILE ON MERGE OUTPUT TAPE
INCREMENT COUNT OF STRINGS ON I/O TAPE
ALL ACTIVE AT START?
YES
7A
10A
NO
PASS
NO. RECS WRITTEN = TOTAL?
NO
ERR9
ERROR
7B
YES
REWIND ALL MERGE TAPES
REVERSE INPUT AND OUTPUT MERGE TAPES
LEVEL COUNTER GREATER THAN 0?
NO
SET LAST PASS SWITCH ON
YES
7


11
EXIT
REWIND MERGE TAPES
11A
SAYSO
WRITE EOF TO WRITE SUBROUTINE
NO OF RECS. WRITTEN = TOTAL ?
NO
ERR7
ERROR
7B
YES
11B
SAVAR
LOAD INDEX REGISTERS
CONTROL PROGRAM


12
WRITE ERROR MESSAGE
LAST PASS?
YES
7B
NO
WAS THIS 2ND TRY AT SORT?
YES
NO
COUNT ANOTHER TRY AT SORT
ONLINE - CHANGE TAPE
HALT
RESTART
1A

COMP SUBROUTINE




ENTER
SAVE XR 1 AND XR4
X = PREVIOUSLY INSERTED RECORD
COMP NEW RECORD SMALLER THAN X?
NO
YES
TOP
INSERT NEW IN CHAIN AS GREATEST MEMBER
NEWGR
X IS GREATEST?
YES
NO
X = FOLLOWER OF X
COMP NEW RECORD SMALLER THAN X?
NO
YES
X = PREDECESSOR OF X
NEWLS
X IS SMALLEST?
YES
NO
BOTM
INSERT NEW IN CHAIN AS SMALLEST MEMBER
COMP NEW RECORD SMALLER THAN X?
YES
NO
NEWER
INSERT NEW IN CHAIN FOLLOWING X
OUT
NEW WILL BE PREVIOUS RECORD NEXT TIME
INSXR
RELOAD XR1 AND XR4
RETURN

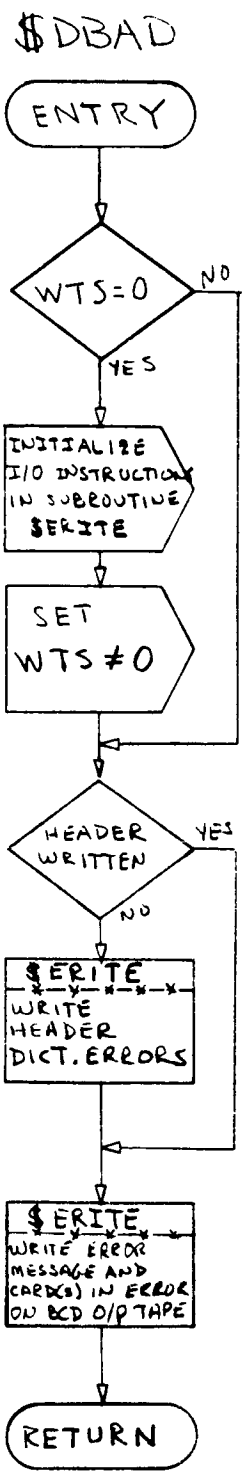
Subroutines UBAD and DBAD

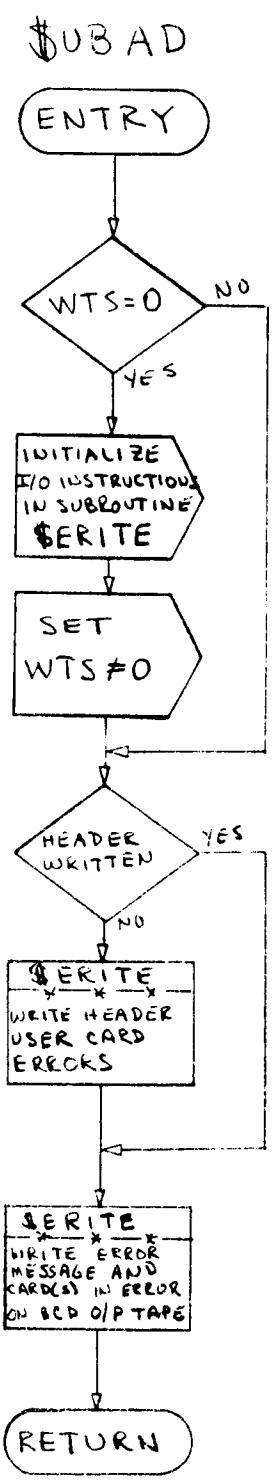
UBAD is entered when an illegal user card is found; DBAD is entered when an illegal vocabulary card is found. Whichever subroutine of these two is entered first will initialize all output instructions in $ERITE.

Every TSX to UBAD or DBAD contains a number in its decrement that determines the error message to be written. This number will be placed in index register one. The RCH instruction effective address of the RCH will point to the I/O command that will write the error message desired.

$DBAD 1/1

$UBAD 1/1





Subroutine UPDATE

Subroutine UPDATE is entered in phase 4. The purpose of UPDATE is to create or update the vocabulary tape while at the same time forming document and/or user profile tapes. This is accomplished by processing information from tape A (input sorted according to coded values).

The operation of the subroutine UPDATE can be discussed for two types of cases: (1) there is no old vocabulary so a new vocabulary is being created from document profile descriptors three words or less in length, or (2) there is an old vocabulary which has to be updated.

Case 1

A calculation of the number of logical records per file is made and blank space is provided for the first file of the new dictionary. Then a record is read from tape A using the subroutine READA.

If the descriptor is illegal it is rejected with an appropriate message on the BCD output tape. All messages on the BCD tape are handled by the subroutine WROUT. For valid descriptors, the type of descriptor is determined and the descriptor is processed accordingly. The possible types of descriptors are as follows:

1. Headers for either documents or users
2. Dictionary changes
    a. Add
    b. Delete
    c. Equate
        i. Primary (T in column 71 of input card)
        ii. Secondary (E in column 71 of input card)
    d. Trouble term
3. Document add descriptors
4. User descriptors
    a. Add
    b. Delete

For either document or user headers the record is sent to the routine MVAD which arranges the record in the format necessary for tapes B and C. Then the record is written out on the appropriate tape (six logical records per block).

For a vocabulary add card, a check is made to see if the descriptor has been added by a previous card. If not, the count of dictionary adds is increased by one, the descriptor is set up in the new dictionary by the subroutine FORM and a BCD message indicating the addition is made on the output tape. The vocabulary tape has a blocking factor of 15, and when the output buffer is full the tape is written by the subroutine WDICT.

If the descriptor is already on the vocabulary, a check is made on the first word of the 20-word alphabetic to see if it is blank. If blank, the alphabetic from the present descriptor is moved into the vocabulary record and a message is printed indicating this; if the alphabetic is not blank, a check is made to determine that the first words of the two alphabetic descriptors match. If they match, a message is printed saying that the descriptor is already on the vocabulary; otherwise a message is printed saying that the coded values are equal but the alphabetics differ. In both cases the descriptor is not added to the vocabulary.

Vocabulary delete records, while valid in an update run, are not allowed in a run to create the vocabulary. A message to this effect is put out on the BCD tape.

For vocabulary equate cards, processing varies depending on whether the descriptor is a primary or a secondary descriptor. If a primary descriptor does not match a descriptor previously added to the vocabulary, the descriptor is set up on the new vocabulary and the count of vocabulary adds is increased by one. A message indicating this action is printed. If there is a match, the descriptor already on the vocabulary is checked to see if it is a base descriptor, and if it is not, a message indicating this is printed. For a secondary descriptor, a check is also made to see if it is already on the vocabulary. If not, the count of vocabulary is increased, a message indicating an addition to the vocabulary is printed and the record is set up on the new vocabulary tape. The second word of the vocabulary record in this case will contain the primary coded value associated with this descriptor and the count of equate cards will be increased by one. If there is a match on the vocabulary, the count of equates will be increased and the primary coded value associated with the descriptor will be moved into the second word of the vocabulary record. A message will be printed indicating that this word has been changed.

User profiles must be corrected to reflect the synonym relationship established by vocabulary equate changes, hence each such change creates a dummy update for user profiles. Thus user profile tapes must be mounted whenever vocabulary equate changes are to be processed. No more than N/2 equate changes should be processed during a single computer run, where N = core storage required by subroutine UPDATE, COMMON storage excluded.

For a trouble term, a check is made to see if the descriptor has already been put on the vocabulary. If not, the count of vocabulary adds is increased, the record is set up on the vocabulary with the second word of the record set equal to zero, and a message indicating the addition of a trouble term is printed. If there is a match, the second word of the vocabulary record is set equal to zero and a message indicating that the descriptor was changed to a trouble term is printed.

For document cards, a check is made to see if the descriptor is on the vocabulary. If not, the descriptor is checked to see if it should be added. If it is not to be added, the descriptor, after being rearranged, is written out on the document tape by the subroutine WDOCT, which uses a blocking factor of six. If the descriptor is to be added to the vocabulary, the count of document adds is increased, a message indicating an addition was made to the vocabulary is printed and the record is set up in the vocabulary. A bit is set in the document record indicating that the descriptor is on the vocabulary, the record is rearranged and is written on the document tape. If the descriptor is already on the vocabulary, the dictionary record is updated and the bit is set in the document record indicating it is on the vocabulary. Then, as in the previous instance, the document record is rearranged and written out onto the document tape.

There are two kinds of user descriptors - user adds and user deletes. A user add is handled

in the same way as a document descriptor except the record is put out onto the user tape by the subroutine WUSE and there is a count for new user additions and a count for additions to old user profiles. User deletes are checked to see if there is a match on the vocabulary. If not, the record is rearranged and written out on the USER tape. If there is a match, the number of users having this descriptor is decreased on the vocabulary record and the record then is handled the same as is a document descriptor having a match on the vocabulary.

Whenever a coded descriptor is to be added to a profile (document or user), the original code is replaced by word 2 of the corresponding vocabulary record. Thus terms previously designated as synonyms or trouble terms are corrected for profile entry (or rejection).

While the descriptor from tape A is processed, an account is kept of the number of descriptors put on the vocabulary so that each file will have approximately 1/100th of the total number of descriptors. Also, a catalog is made of the minimum and maximum coded values for each file. In the situation where 100 files have not been written and tape A is finished, the remaining files will each contain 450 words of zeros and have maximum and minimum values consisting of computer words of octal sevens. When all descriptors have been processed the vocabulary tape is rewound and the first file record contains the tape label. The second record contains the catalog of minimum and maximum entries and the third record has a table of entries containing the number of logical records per file. This file is written by the subroutine WDLBL. Also, if there is anything in the buffers for the document and user tapes, these tapes are written. Finally, a summary of all the changes made to the vocabulary is written on the system output tape.

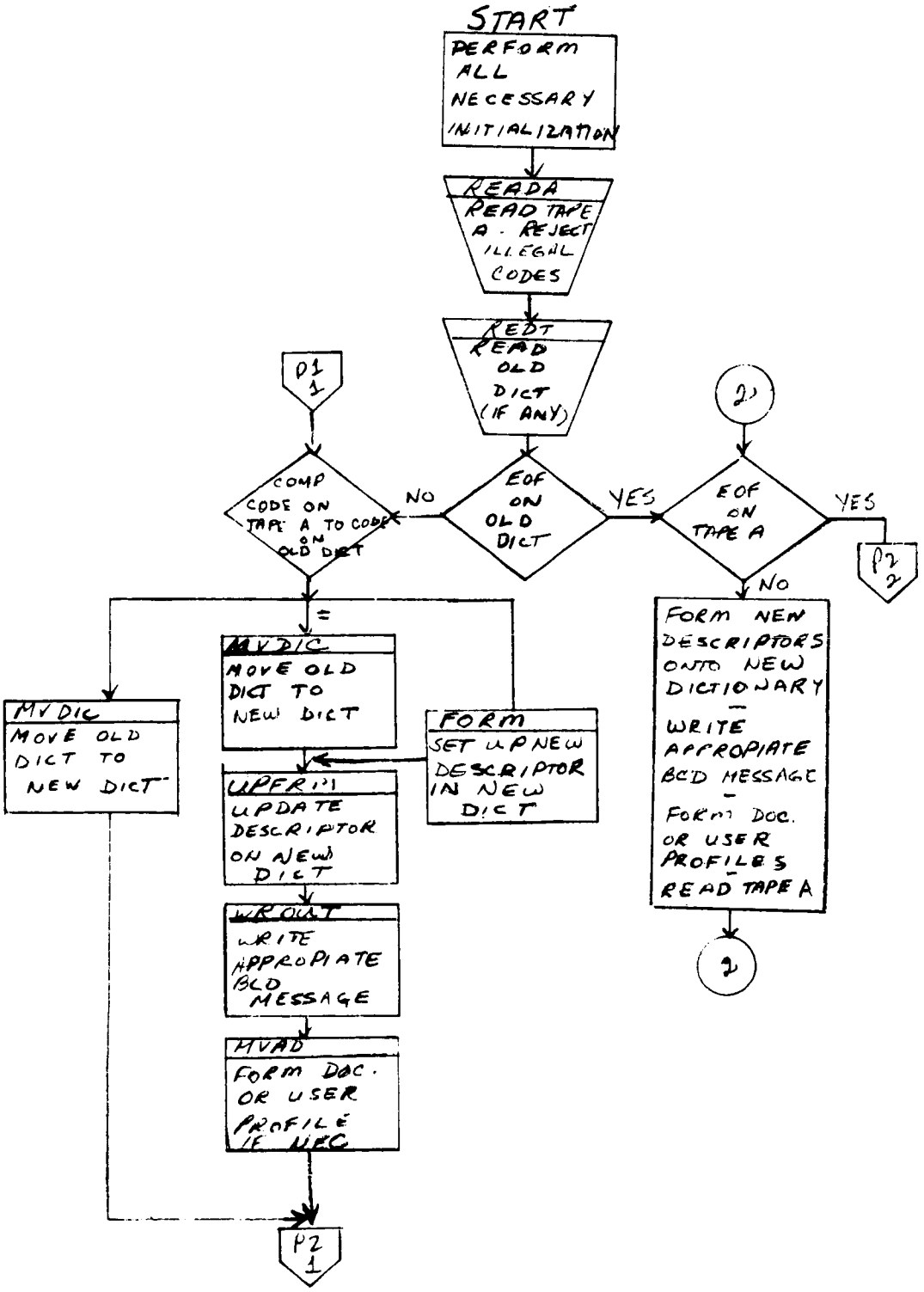
Case 2

In a run to update the vocabulary, all descriptor records have to be matched against the old vocabulary. Until a match is found, the records read from the old vocabulary by the subroutine REDT are written out onto the new vocabulary. The size of each file in the new vocabulary will depend, of course, on both the number of descriptors on tape A and the number of descriptors on the old vocabulary. Also, the maximum and minimum coded values are again provided for each file on the new vocabulary.

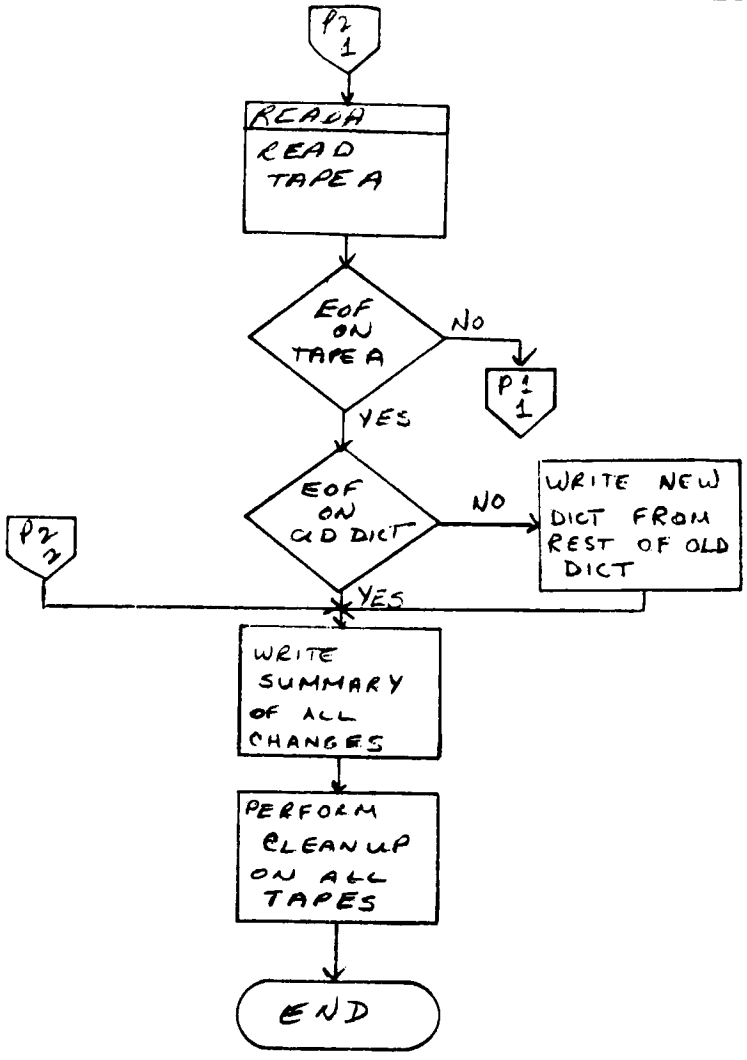
In general, for an updating run, the types of descriptors are processed in the same way as in a run to create the vocabulary, except that dictionary delete descriptors are allowed. The descriptor is checked for a match on the old vocabulary or for a match with a descriptor just added to the vocabulary. If there is no match, a message indicating that the descriptor is not on the vocabulary is printed. If there is a match, the count of vocabulary deletes is increased, the record is deleted from the vocabulary and a message is printed saying the descriptor has been deleted.

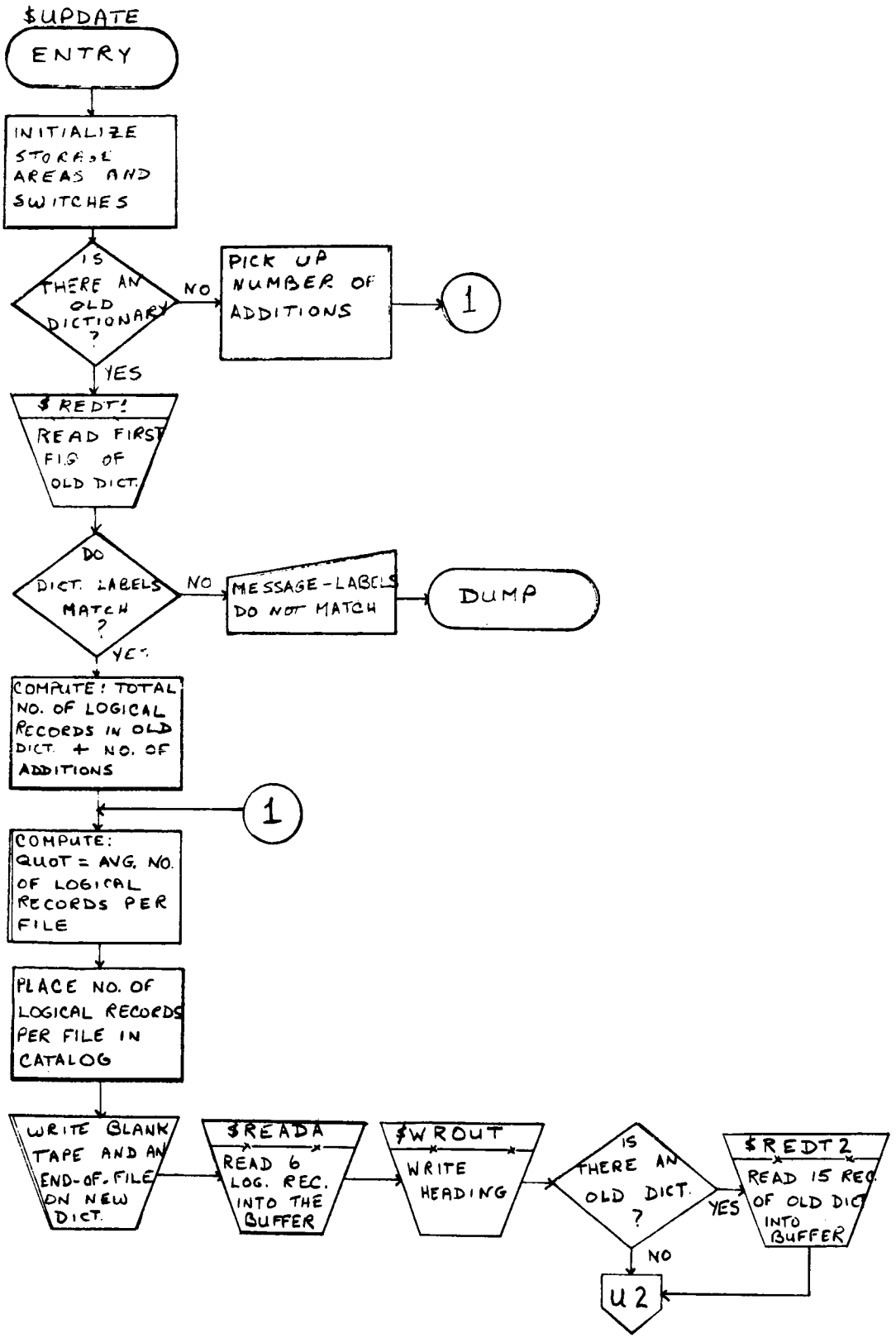
If all processing of tape A is done but there still remain records on the old vocabulary, these records will be written from the old vocabulary onto the new vocabulary. On the other hand, if the old vocubulary is finished before tape A, the remaining descriptors will be processed from tape A as described in Case 1. In either situation, final processing of the new vocabulary tape and the document and user tapes is accomplished in the same manner as in Case 1.

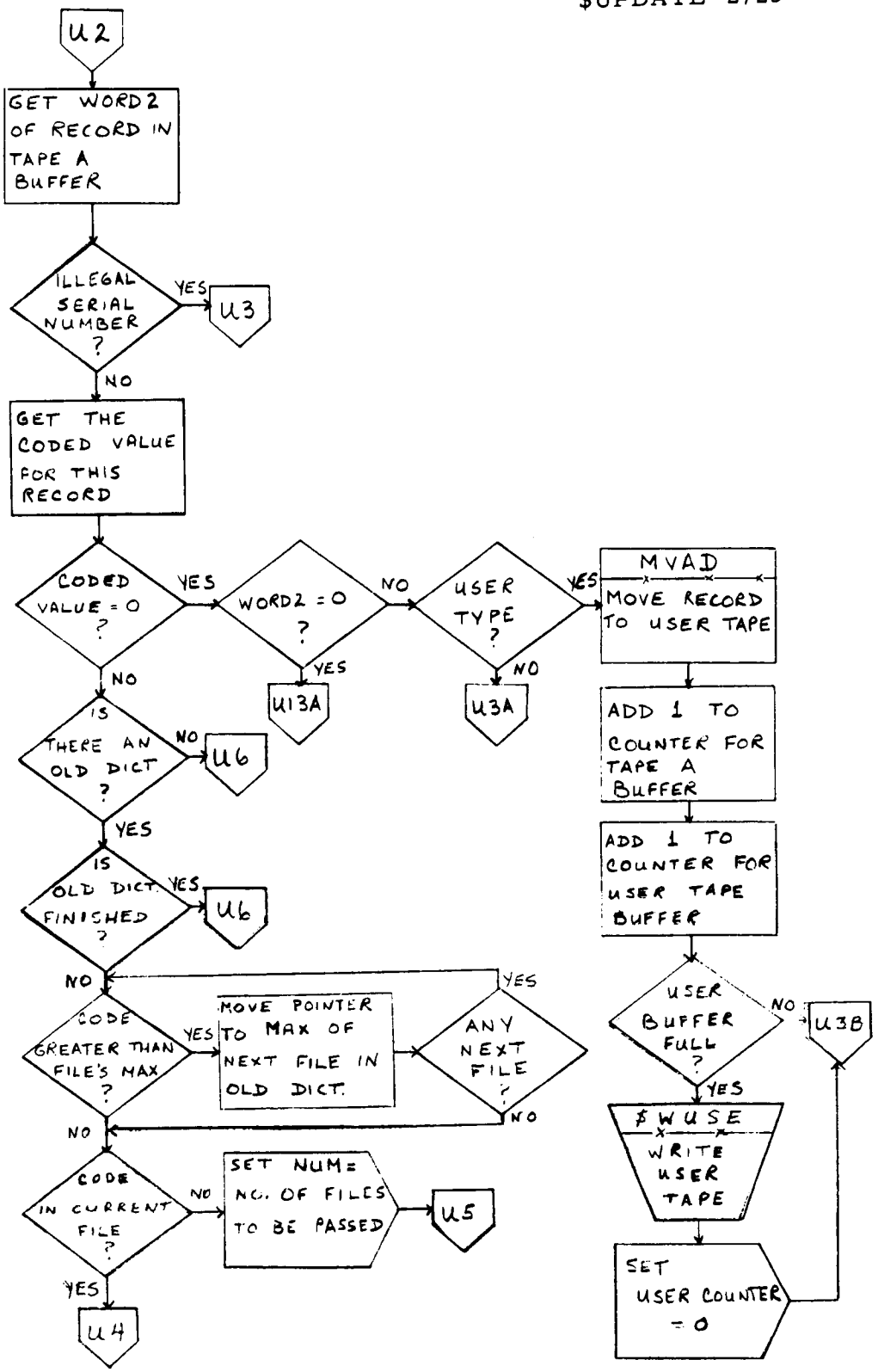
Block Diagram

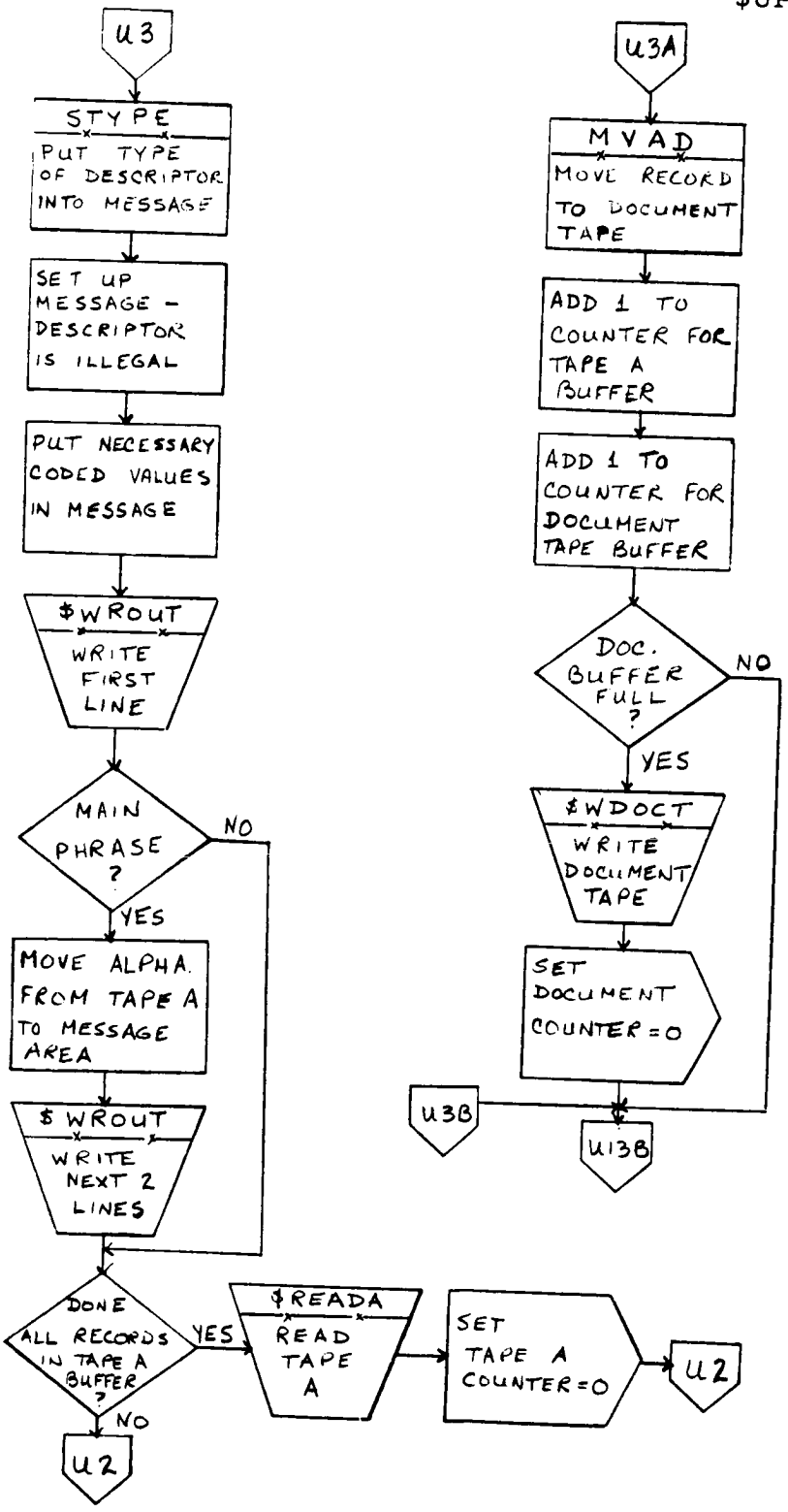


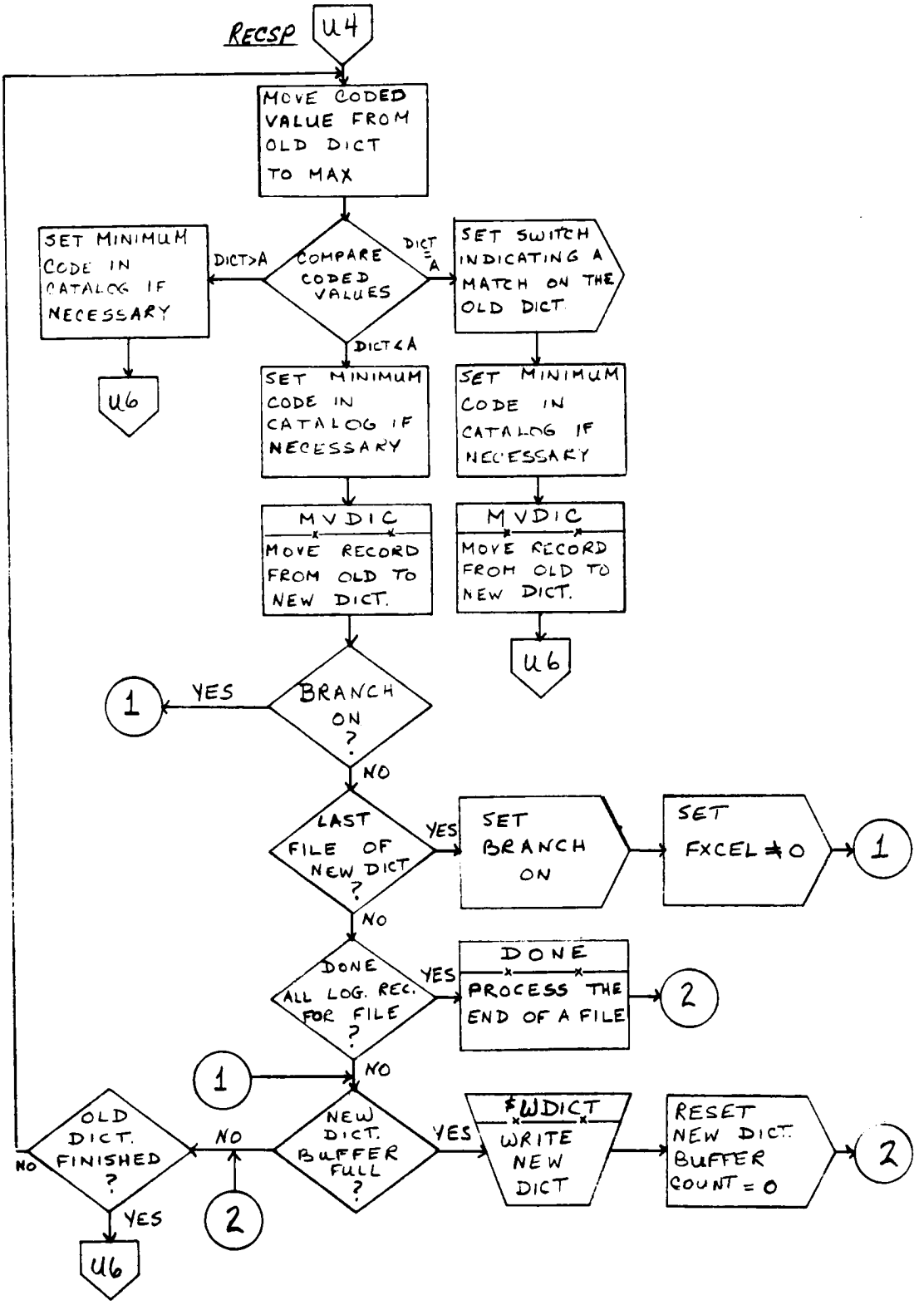
Block Diagram

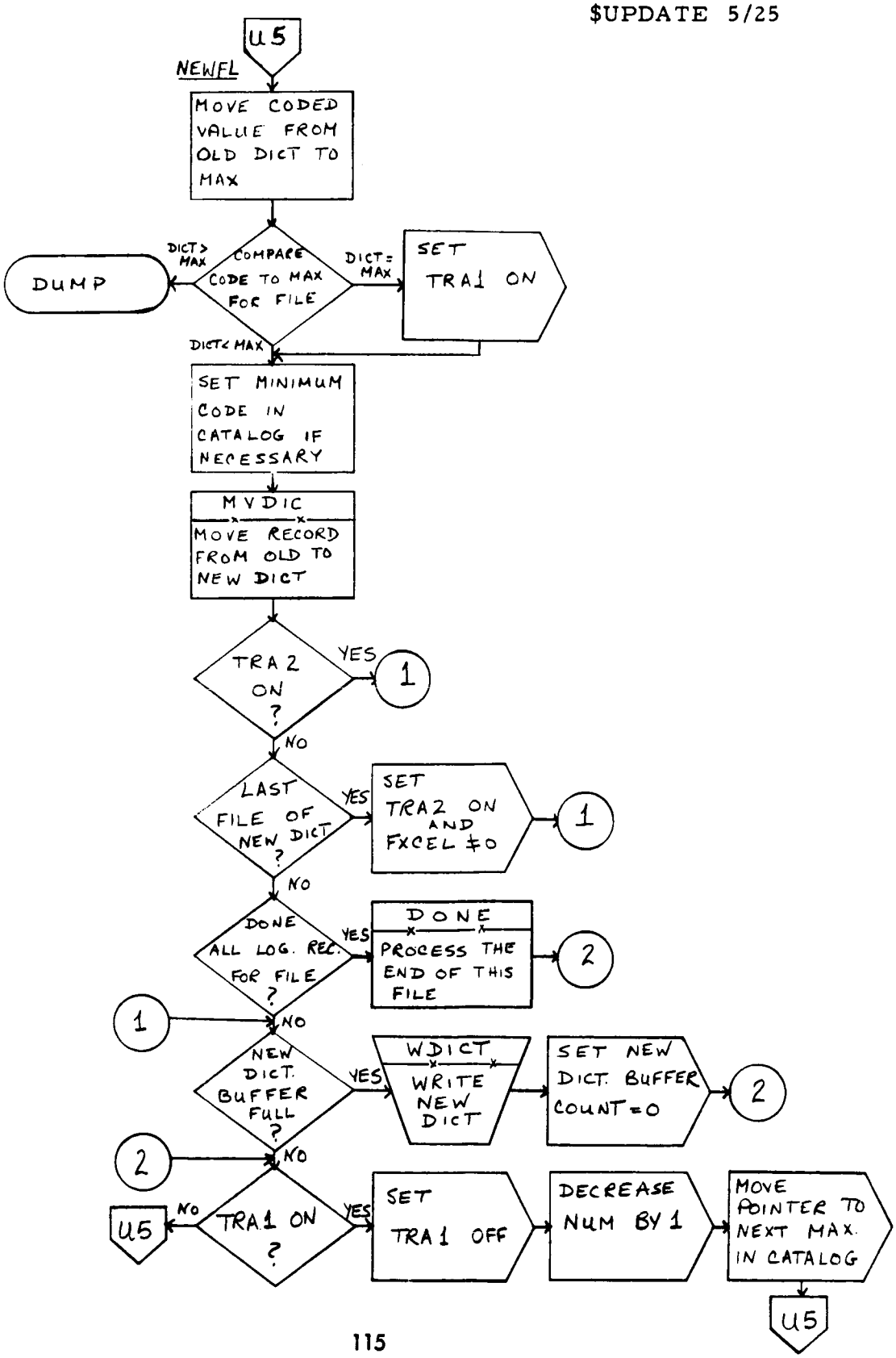


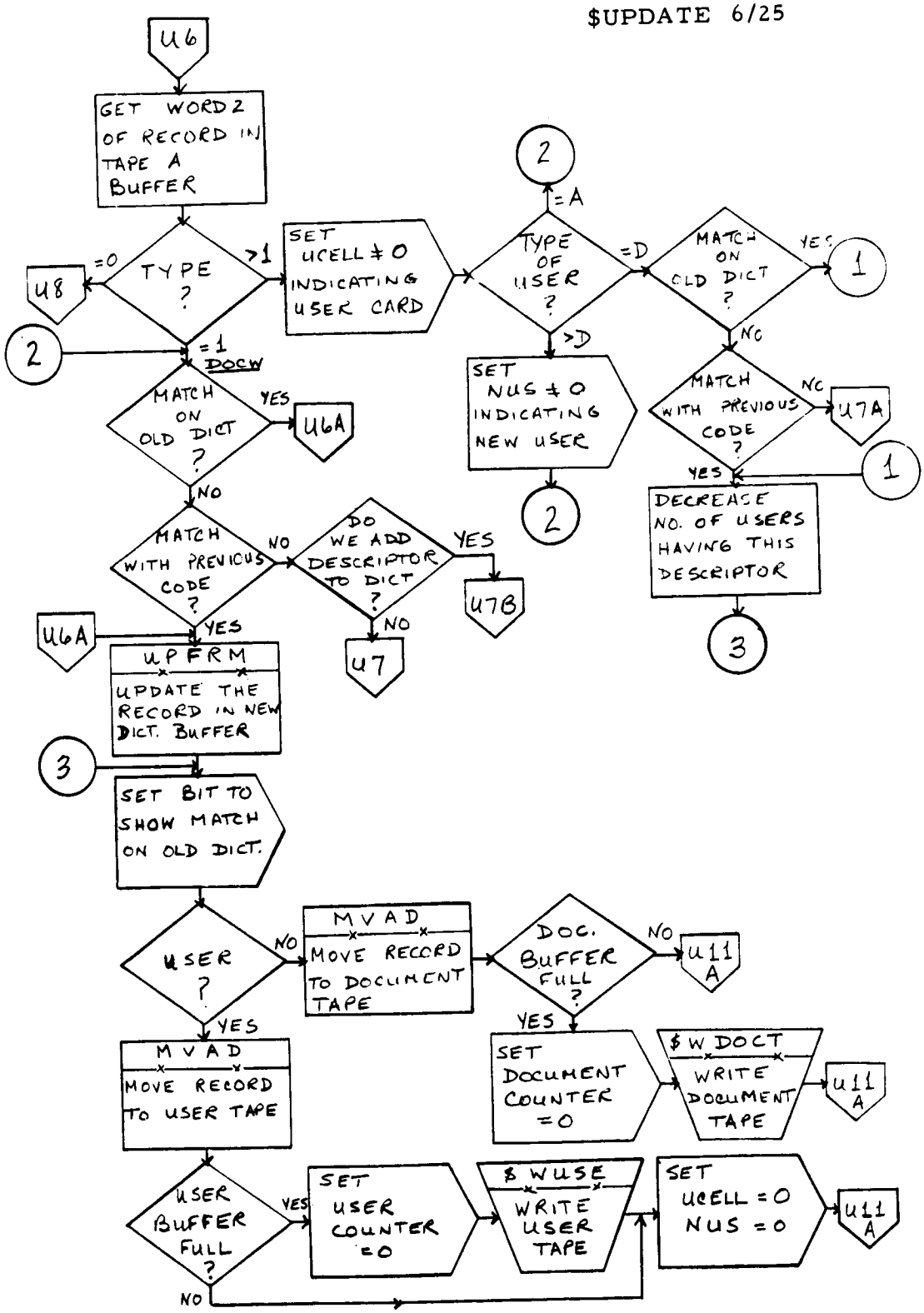

$UPDATE
ENTRY
INITIALIZE STORAGE AREAS AND SWITCHES
IS THERE AN OLD DICTIONARY ?
NO
PICK UP NUMBER OF ADDITIONS
1
YES
$REDT1
READ FIRST FIL OF OLD DICT.
DO DICT. LABELS MATCH ?
NO
MESSAGE-LABELS DO NOT MATCH
DUMP
YES
COMPUTE: TOTAL NO. OF LOGICAL RECORDS IN OLD DICT. + NO. OF ADDITIONS
1
COMPUTE: QUOT = AVG. NO. OF LOGICAL RECORDS PER FILE
PLACE NO. OF LOGICAL RECORDS PER FILE IN CATALOG
WRITE BLANK TAPE AND AN END-OF-FILE ON NEW DICT.
$READA
READ 6 LOG. REC. INTO THE BUFFER
$WROUT
WRITE HEADING
IS THERE AN OLD DICT. ?
YES
$REDT2
READ 15 REC OF OLD DICT INTO BUFFER
NO
U2

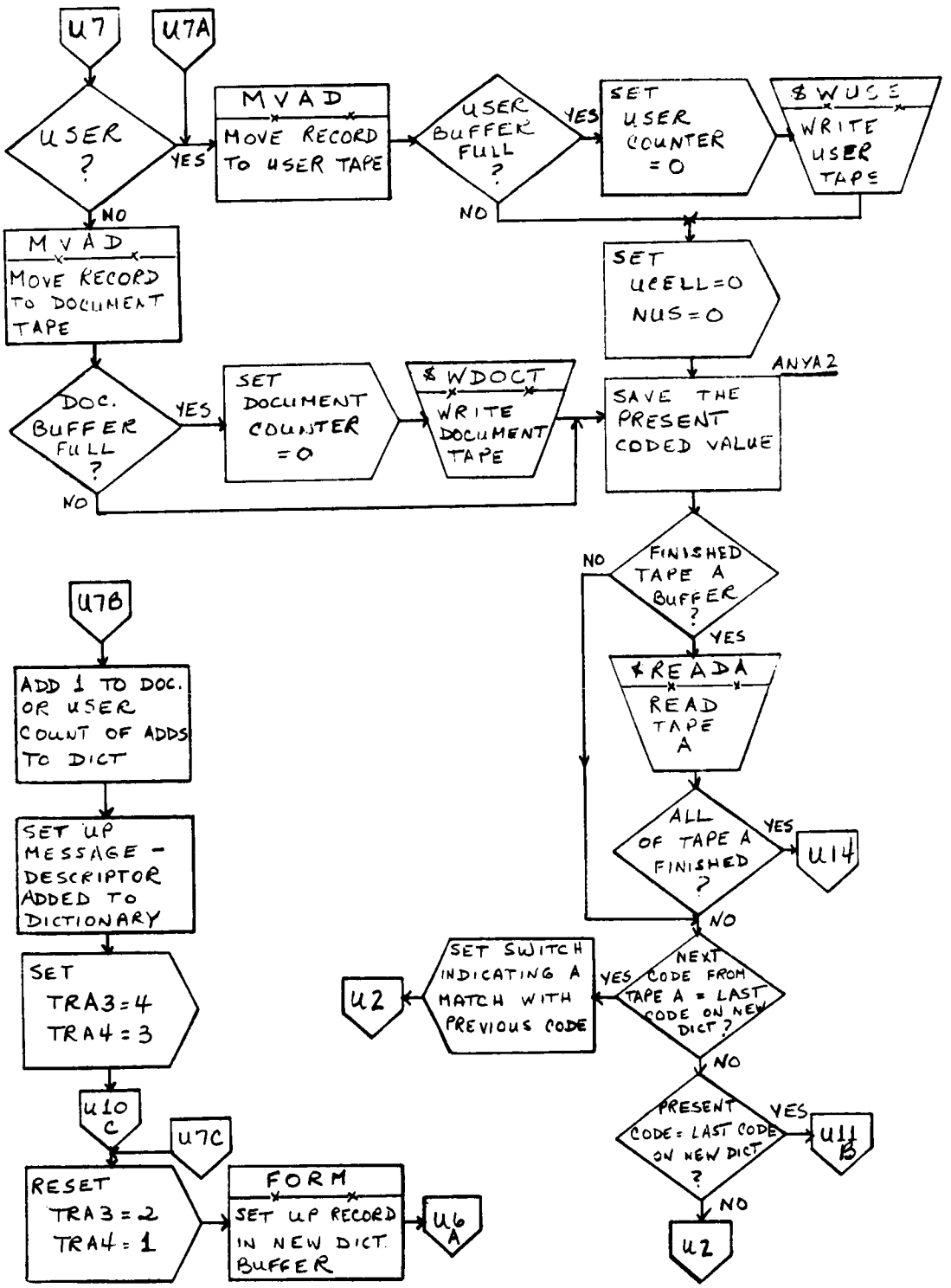

U2
GET WORD2 OF RECORD IN TAPE A BUFFER
ILLEGAL SERIAL NUMBER ?
YES
U3
NO
GET THE CODED VALUE FOR THIS RECORD
CODED VALUE = 0 ?
YES
WORD2 = 0 ?
NO
USER TYPE ?
YES
MVAD
MOVE RECORD TO USER TAPE
NO
U13A
NO
U3A
ADD 1 TO COUNTER FOR TAPE A BUFFER
IS THERE AN OLD DICT ?
NO
U6
YES
ADD 1 TO COUNTER FOR USER TAPE BUFFER
IS OLD DICT FINISHED ?
YES
U6
NO
YES
USER BUFFER FULL ?
NO
U3B
CODE GREATER THAN FILE'S MAX ?
YES
MOVE POINTER TO MAX OF NEXT FILE IN OLD DICT.
ANY NEXT FILE ?
YES
NO
YES
$WUSE
WRITE USER TAPE
NO
CODE IN CURRENT FILE ?
NO
SET NUM= NO. OF FILES TO BE PASSED
U5
YES
U4
SET USER COUNTER = 0

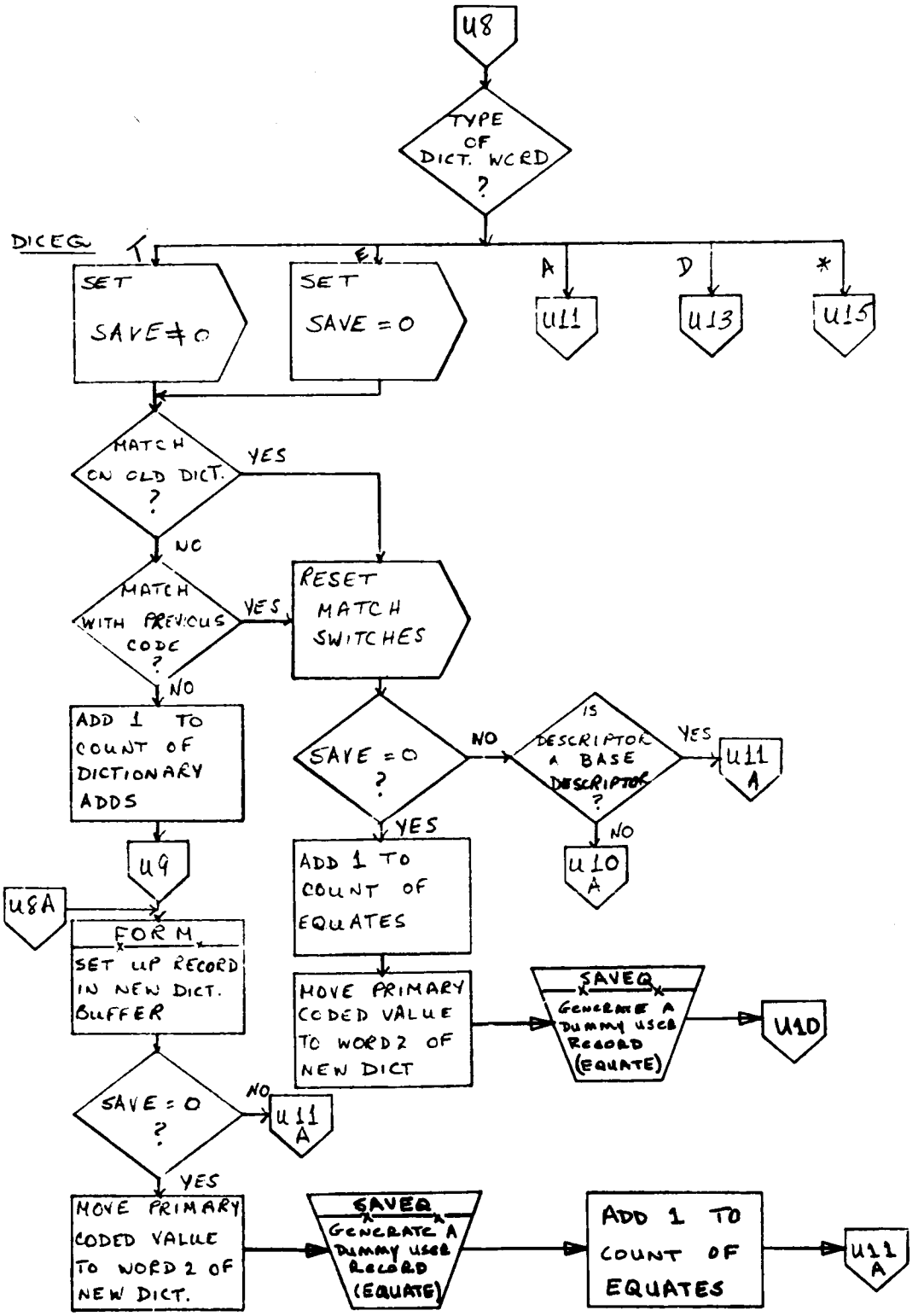

U3
STYPE
PUT TYPE OF DESCRIPTOR INTO MESSAGE
SET UP MESSAGE - DESCRIPTOR IS ILLEGAL
PUT NECESSARY CODED VALUES IN MESSAGE
$WROUT
WRITE FIRST LINE
MAIN PHRASE ?
NO
YES
MOVE ALPHA FROM TAPE A TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
DONE ALL RECORDS IN TAPE A BUFFER ?
YES
NO
U2
$READA
READ TAPE A
SET TAPE A COUNTER=0
U2
U3A
MVAD
MOVE RECORD TO DOCUMENT TAPE
ADD 1 TO COUNTER FOR TAPE A BUFFER
ADD 1 TO COUNTER FOR DOCUMENT TAPE BUFFER
DOC. BUFFER FULL ?
NO
YES
$WDOCT
WRITE DOCUMENT TAPE
SET DOCUMENT COUNTER=0
U3B
U13B

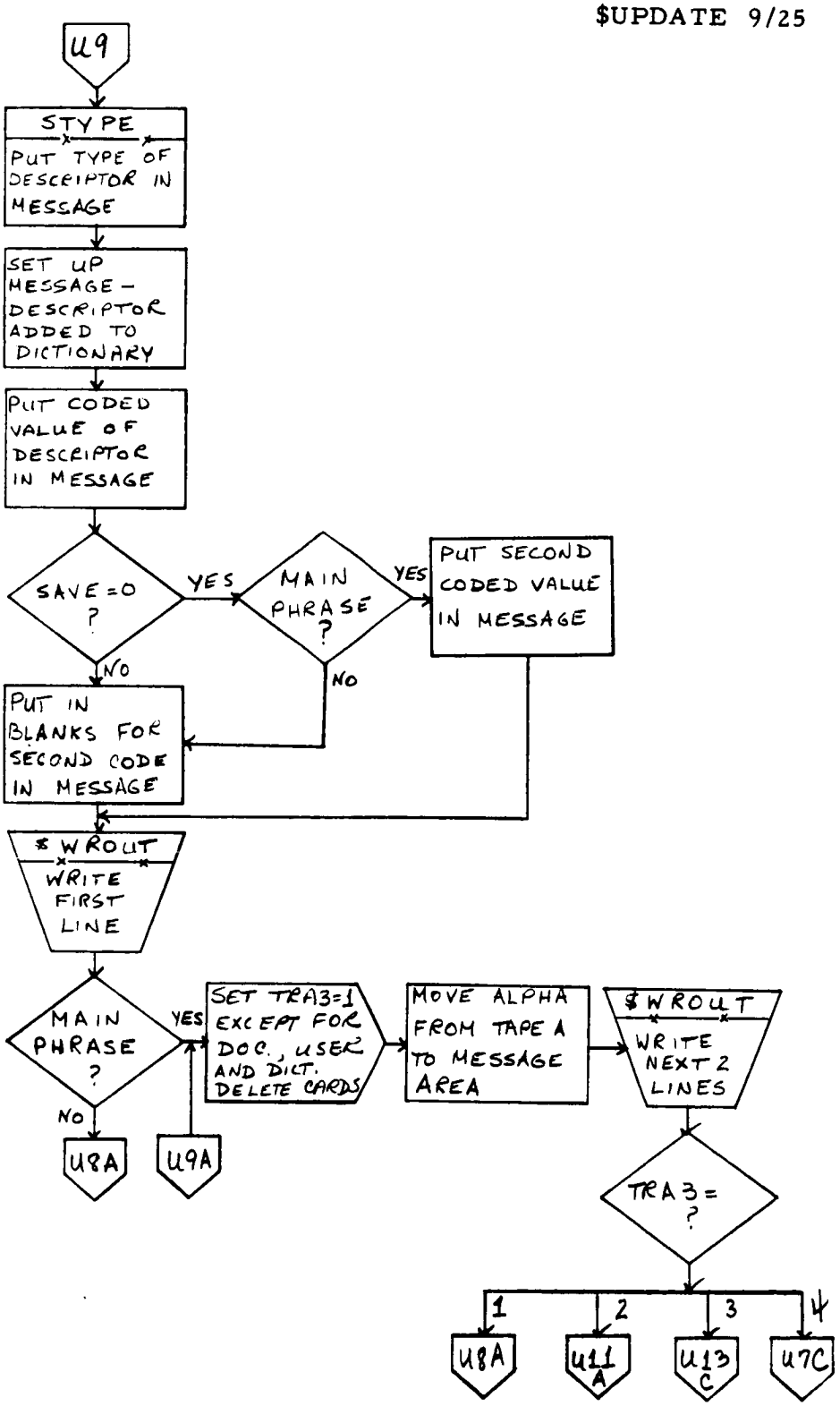

RECSP
U4
MOVE CODED VALUE FROM OLD DICT TO MAX
COMPARE CODED VALUES
DICT>A
DICT=A
DICT<A
SET MINIMUM CODE IN CATALOG IF NECESSARY
U6
SET SWITCH INDICATING A MATCH ON THE OLD DICT
SET MINIMUM CODE IN CATALOG IF NECESSARY
MVDIC
MOVE RECORD FROM OLD TO NEW DICT.
MVDIC
MOVE RECORD FROM OLD TO NEW DICT.
U6
BRANCH ON ?
YES
1
NO
LAST FILE OF NEW DICT ?
YES
SET BRANCH ON
SET FXCEL ≠ 0
1
NO
DONE ALL LOG. REC. FOR FILE ?
YES
DONE
PROCESS THE END OF A FILE
2
NO
1
NEW DICT. BUFFER FULL ?
YES
$WDICT
WRITE NEW DICT
RESET NEW DICT. BUFFER COUNT = 0
2
NO
OLD DICT. FINISHED ?
NO
YES
U6
2

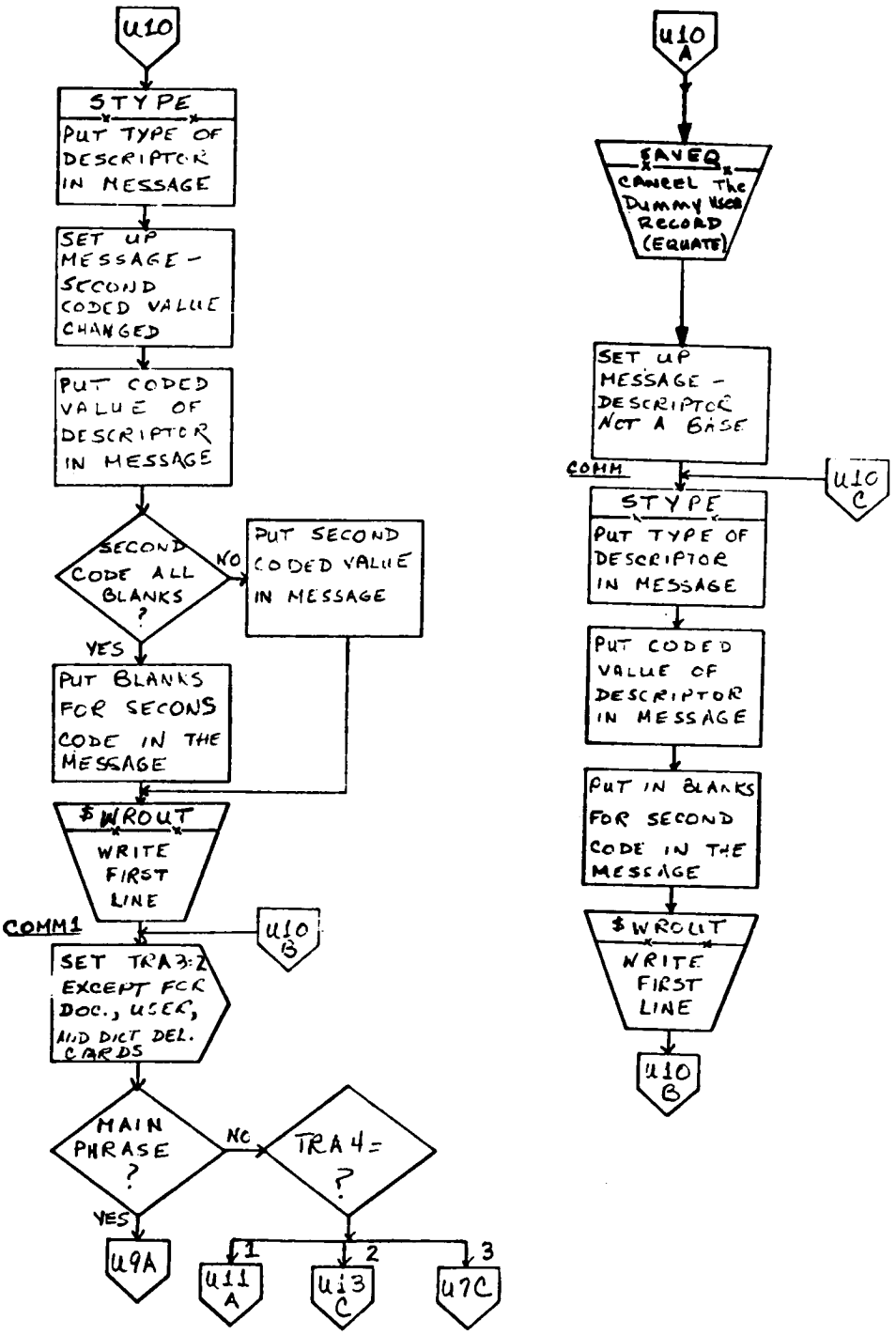

U5
NEWFL
MOVE CODED VALUE FROM OLD DICT TO MAX
COMPARE CODE TO MAX FOR FILE
DICT > MAX
DUMP
DICT = MAX
SET TRA1 ON
DICT < MAX
SET MINIMUM CODE IN CATALOG IF NECESSARY
MVDIC
MOVE RECORD FROM OLD TO NEW DICT
TRA 2 ON ?
YES
1
NO
LAST FILE OF NEW DICT ?
YES
SET TRA2 ON AND FXCEL ≠0
1
NO
DONE ALL LOG. REC. FOR FILE ?
YES
DONE
PROCESS THE END OF THIS FILE
2
1
NO
NEW DICT. BUFFER FULL ?
YES
WDICT
WRITE NEW DICT
SET NEW DICT. BUFFER COUNT=0
2
2
NO
U5
NO
TRA1 ON ?
YES
SET TRA1 OFF
DECREASE NUM BY 1
MOVE POINTER TO NEXT MAX. IN CATALOG
U5

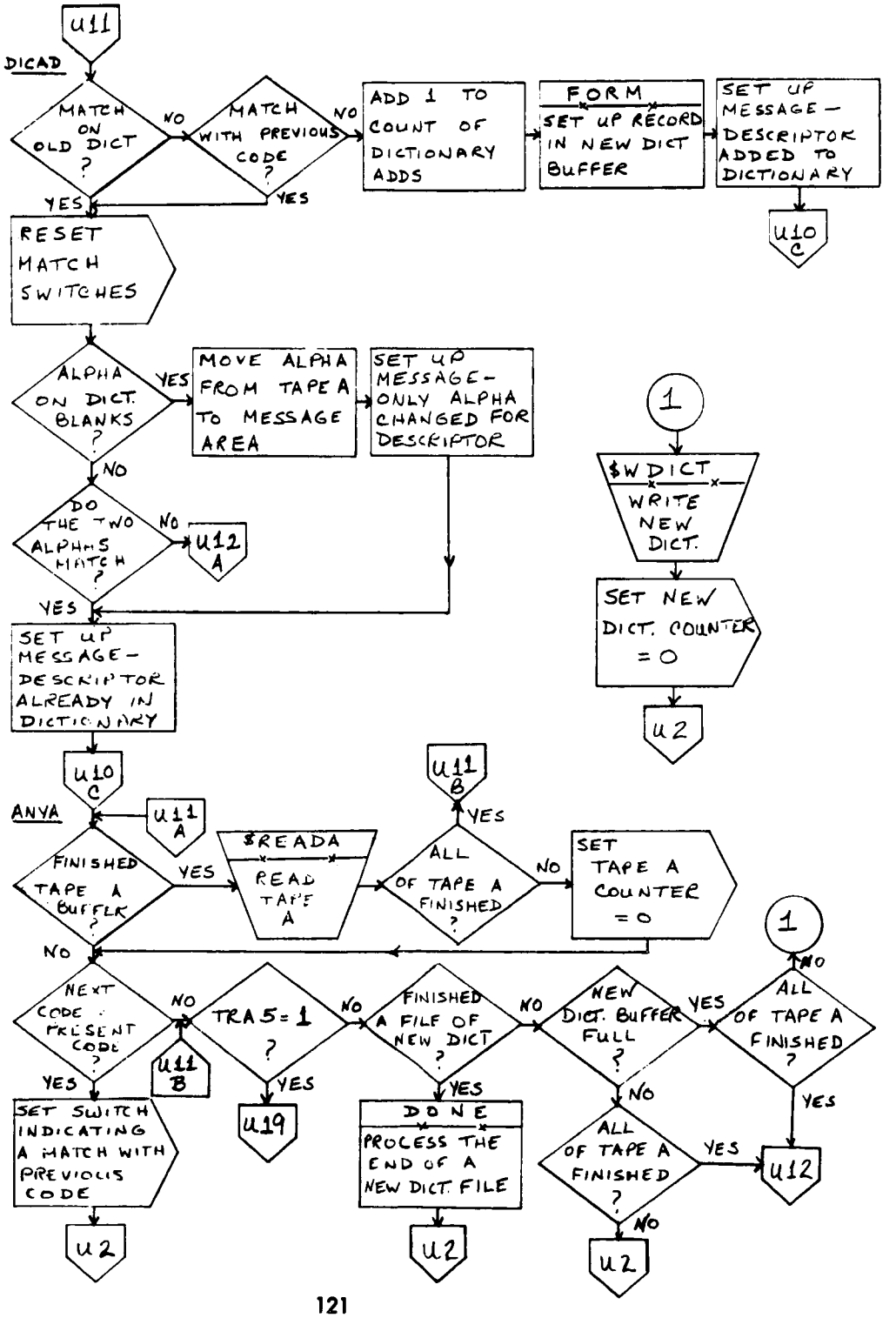

U6
GET WORD 2 OF RECORD IN TAPE A BUFFER
TYPE ?
=0
>1
U8
SET UCELL ≠ 0 INDICATING USER CARD
TYPE OF USER ?
=A
=D
>D
MATCH ON OLD DICT ?
YES
NO
1
SET NUS ≠ 0 INDICATING NEW USER
MATCH WITH PREVIOUS CODE ?
NO
U7A
YES
DECREASE NO. OF USERS HAVING THIS DESCRIPTOR
3
2
=1 DOCW
MATCH ON OLD DICT ?
YES
U6A
NO
MATCH WITH PREVIOUS CODE ?
NO
DO WE ADD DESCRIPTOR TO DICT ?
YES
U7B
NO
U7
U6A
YES
UPFRM
UPDATE THE RECORD IN NEW DICT. BUFFER
3
SET BIT TO SHOW MATCH ON OLD DICT.
USER ?
NO
MVAD
MOVE RECORD TO DOCUMENT TAPE
DOC. BUFFER FULL ?
NO
U11 A
YES
SET DOCUMENT COUNTER =0
$WDOCT
WRITE DOCUMENT TAPE
U11 A
YES
MVAD
MOVE RECORD TO USER TAPE
USER BUFFER FULL ?
YES
SET USER COUNTER =0
$WUSE
WRITE USER TAPE
SET UCELL = 0 NUS = 0
U11 A
NO

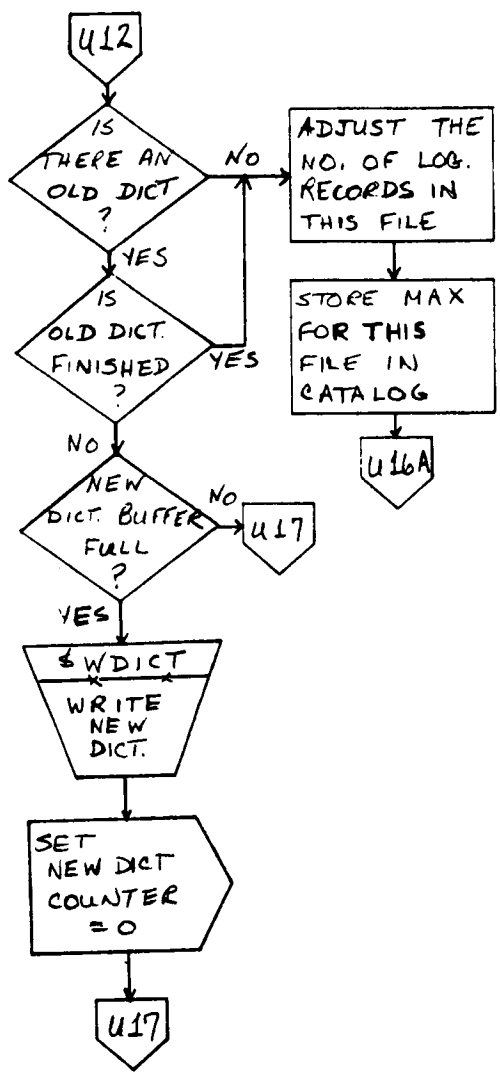

U7
U7A
USER ?
YES
NO
MVAD
MOVE RECORD TO USER TAPE
USER BUFFER FULL ?
YES
NO
SET USER COUNTER = 0
$WUSE
WRITE USER TAPE
MVAD
MOVE RECORD TO DOCUMENT TAPE
SET UCELL=0 NUS=0
ANYA2
DOC. BUFFER FULL ?
YES
NO
SET DOCUMENT COUNTER = 0
$WDOCT
WRITE DOCUMENT TAPE
SAVE THE PRESENT CODED VALUE
FINISHED TAPE A BUFFER ?
NO
YES
$READA
READ TAPE A
ALL OF TAPE A FINISHED ?
YES
U14
NO
U7B
ADD 1 TO DOC. OR USER COUNT OF ADDS TO DICT
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
SET TRA3=4 TRA4=3
U10 C
U7C
RESET TRA3=2 TRA4=1
FORM
SET UP RECORD IN NEW DICT BUFFER
U6 A
U2
SET SWITCH INDICATING A MATCH WITH PREVIOUS CODE
YES
NEXT CODE FROM TAPE A = LAST CODE ON NEW DICT ?
NO
PRESENT CODE = LAST CODE ON NEW DICT ?
YES
U11 B
NO
U2

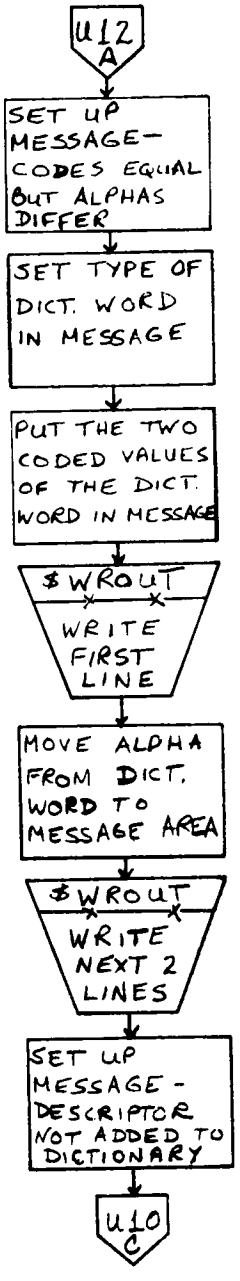

U8
TYPE OF DICT. WORD ?
DICEQ
T
F
A
D
*
U11
U13
U15
SET SAVE ≠ 0
SET SAVE = 0
MATCH ON OLD DICT. ?
YES
NO
MATCH WITH PREVIOUS CODE ?
YES
NO
RESET MATCH SWITCHES
ADD 1 TO COUNT OF DICTIONARY ADDS
U9
U8A
FORM
SET UP RECORD IN NEW DICT. BUFFER
SAVE = 0 ?
NO
U11 A
YES
MOVE PRIMARY CODED VALUE TO WORD 2 OF NEW DICT.
SAVEQ
GENERATE A DUMMY USER RECORD (EQUATE)
ADD 1 TO COUNT OF EQUATES
U11 A
SAVE = 0 ?
NO
YES
IS DESCRIPTOR A BASE DESCRIPTOR ?
YES
NO
U11 A
U10 A
ADD 1 TO COUNT OF EQUATES
MOVE PRIMARY CODED VALUE TO WORD 2 OF NEW DICT
SAVEQ
GENERATE A DUMMY USER RECORD (EQUATE)
U10

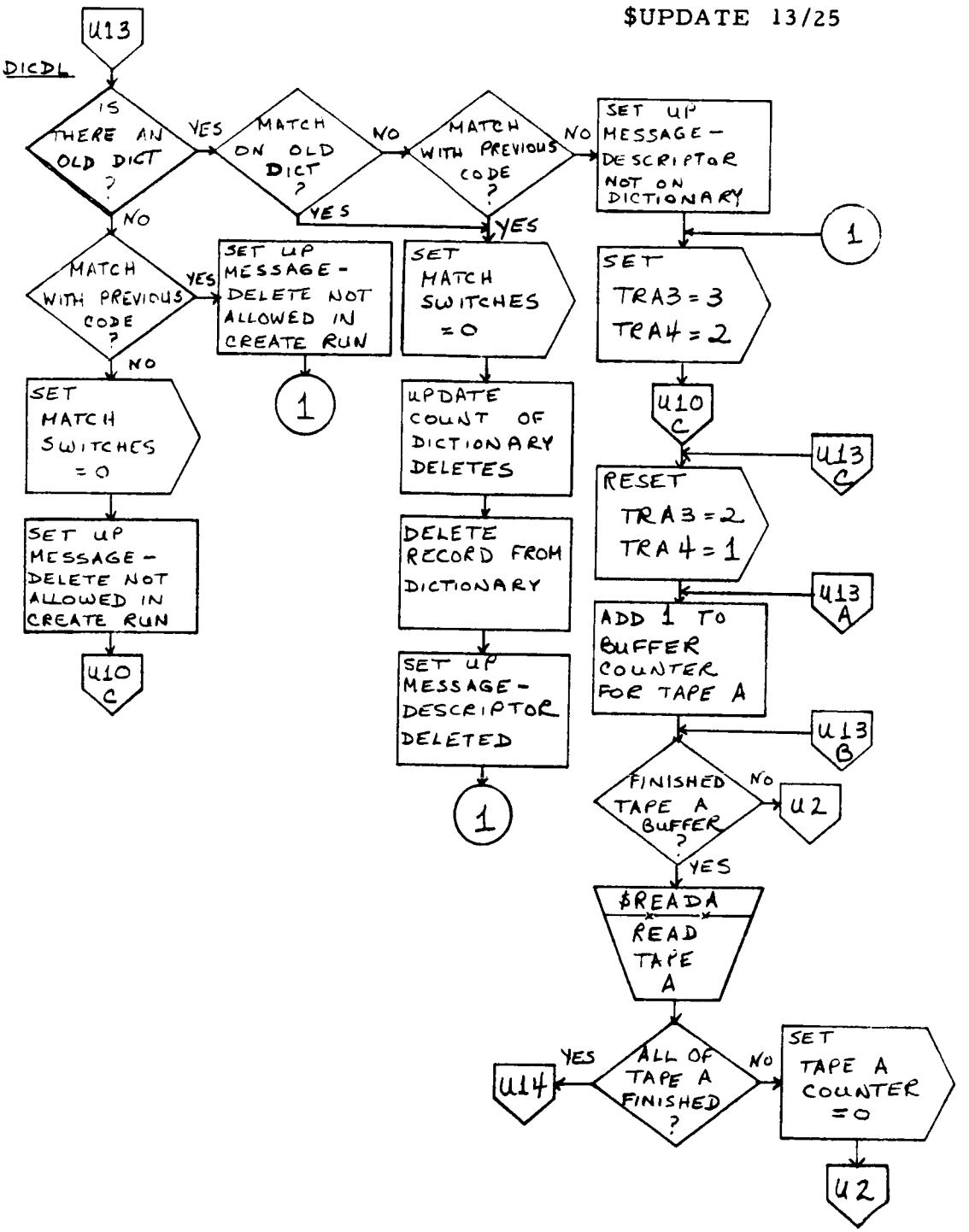

U9
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
SAVE=0 ?
YES
MAIN PHRASE ?
YES
PUT SECOND CODED VALUE IN MESSAGE
NO
NO
PUT IN BLANKS FOR SECOND CODE IN MESSAGE
$WROUT
WRITE FIRST LINE
MAIN PHRASE ?
YES
SET TRA3=1 EXCEPT FOR DOC., USER AND DICT. DELETE CARDS
MOVE ALPHA FROM TAPE A TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
NO
U8A
U9A
TRA3 = ?
1
2
3
4
U8A
U11A
U13C
U7C

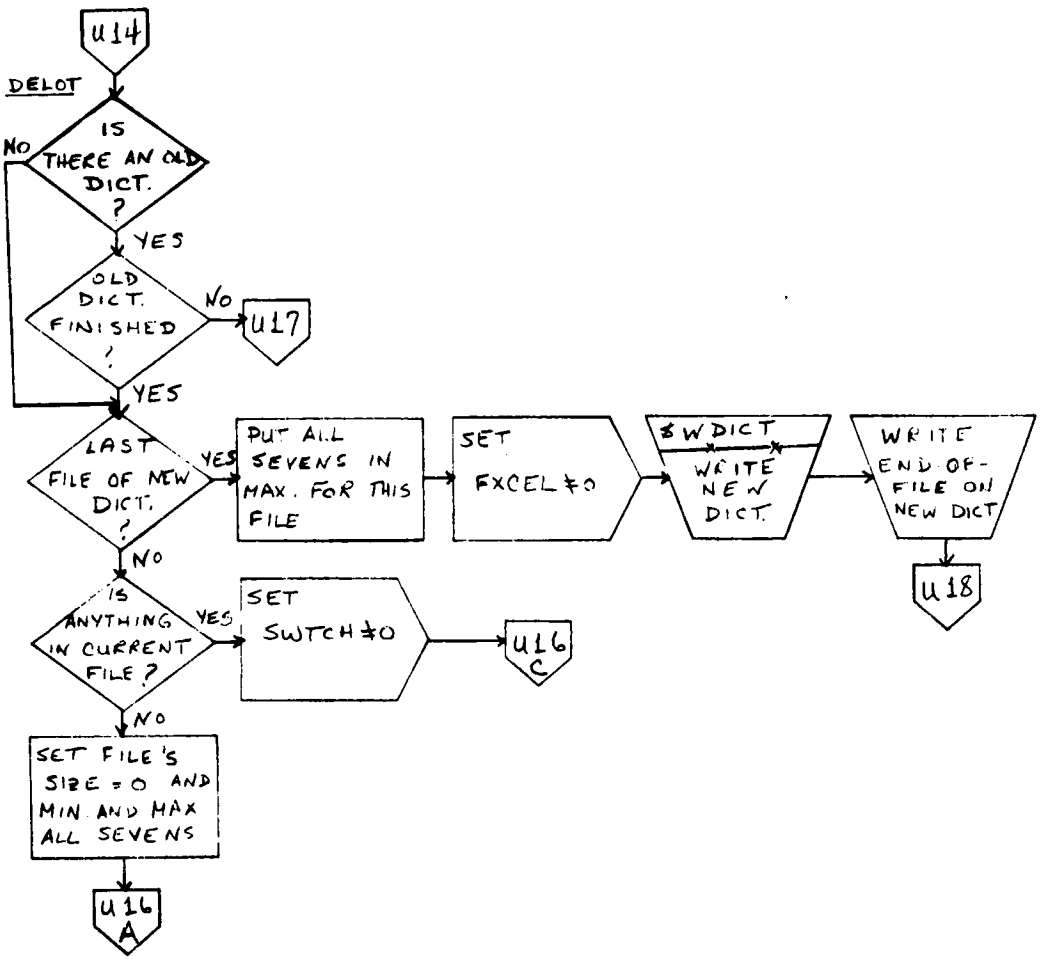

U10
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
SET UP MESSAGE - SECOND CODED VALUE CHANGED
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
SECOND CODE ALL BLANKS ?
NO
PUT SECOND CODED VALUE IN MESSAGE
YES
PUT BLANKS FOR SECONS CODE IN THE MESSAGE
$WROUT
WRITE FIRST LINE
COMM1
U10 B
SET TRA3:2 EXCEPT FOR DOC., USER, AND DICT DEL. CARDS
MAIN PHRASE ?
NO
TRA4= ?
YES
U9A
1
2
3
U11 A
U13 C
U7C
U10 A
$AVEQ
CANCEL THE DUMMY USER RECORD (EQUATE)
SET UP MESSAGE - DESCRIPTOR NOT A BASE
COMM
U10 C
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
PUT IN BLANKS FOR SECOND CODE IN THE MESSAGE
$WROUT
WRITE FIRST LINE
U10 B

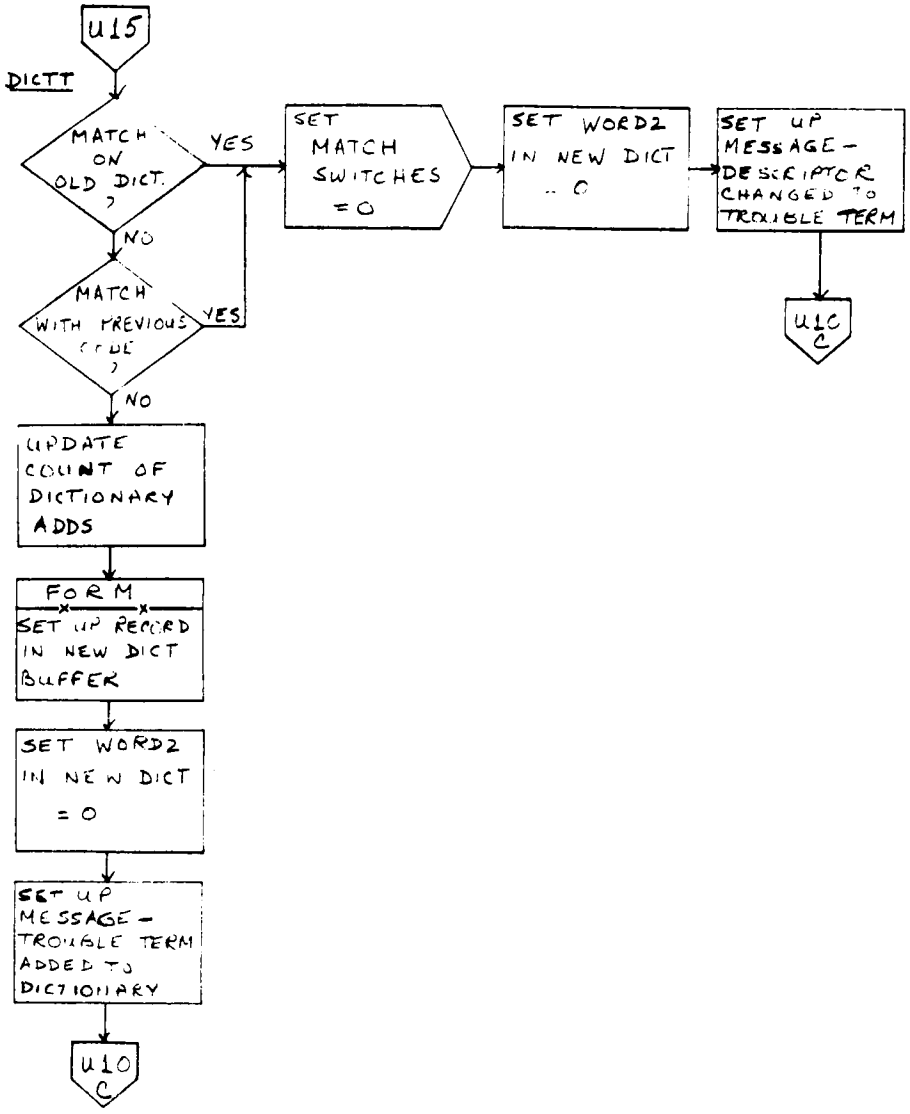

u11
DICAD
MATCH ON OLD DICT ?
NO
MATCH WITH PREVIOUS CODE ?
NO
YES
YES
ADD 1 TO COUNT OF DICTIONARY ADDS
FORM
SET UP RECORD IN NEW DICT BUFFER
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
u10 C
RESET MATCH SWITCHES
ALPHA ON DICT. BLANKS ?
YES
MOVE ALPHA FROM TAPE A TO MESSAGE AREA
SET UP MESSAGE - ONLY ALPHA CHANGED FOR DESCRIPTOR
NO
DO THE TWO ALPHAS MATCH ?
NO
u12 A
YES
SET UP MESSAGE - DESCRIPTOR ALREADY IN DICTIONARY
1
$W DICT
WRITE NEW DICT.
SET NEW DICT. COUNTER = 0
u2
u10 C
ANYA
u11 A
FINISHED TAPE A BUFFER ?
YES
$READA
READ TAPE A
u11 B
YES
ALL OF TAPE A FINISHED ?
NO
SET TAPE A COUNTER = 0
NO
NEXT CODE = PRESENT CODE ?
NO
TRA 5 = 1 ?
NO
FINISHED A FILE OF NEW DICT ?
NO
NEW DICT. BUFFER FULL ?
YES
ALL OF TAPE A FINISHED ?
NO
1
YES
u11 B
YES
u19
YES
DONE
PROCESS THE END OF A NEW DICT FILE
YES
NO
ALL OF TAPE A FINISHED ?
YES
u12
YES
SET SWITCH INDICATING A MATCH WITH PREVIOUS CODE
u2
u2
u2
NO
u2

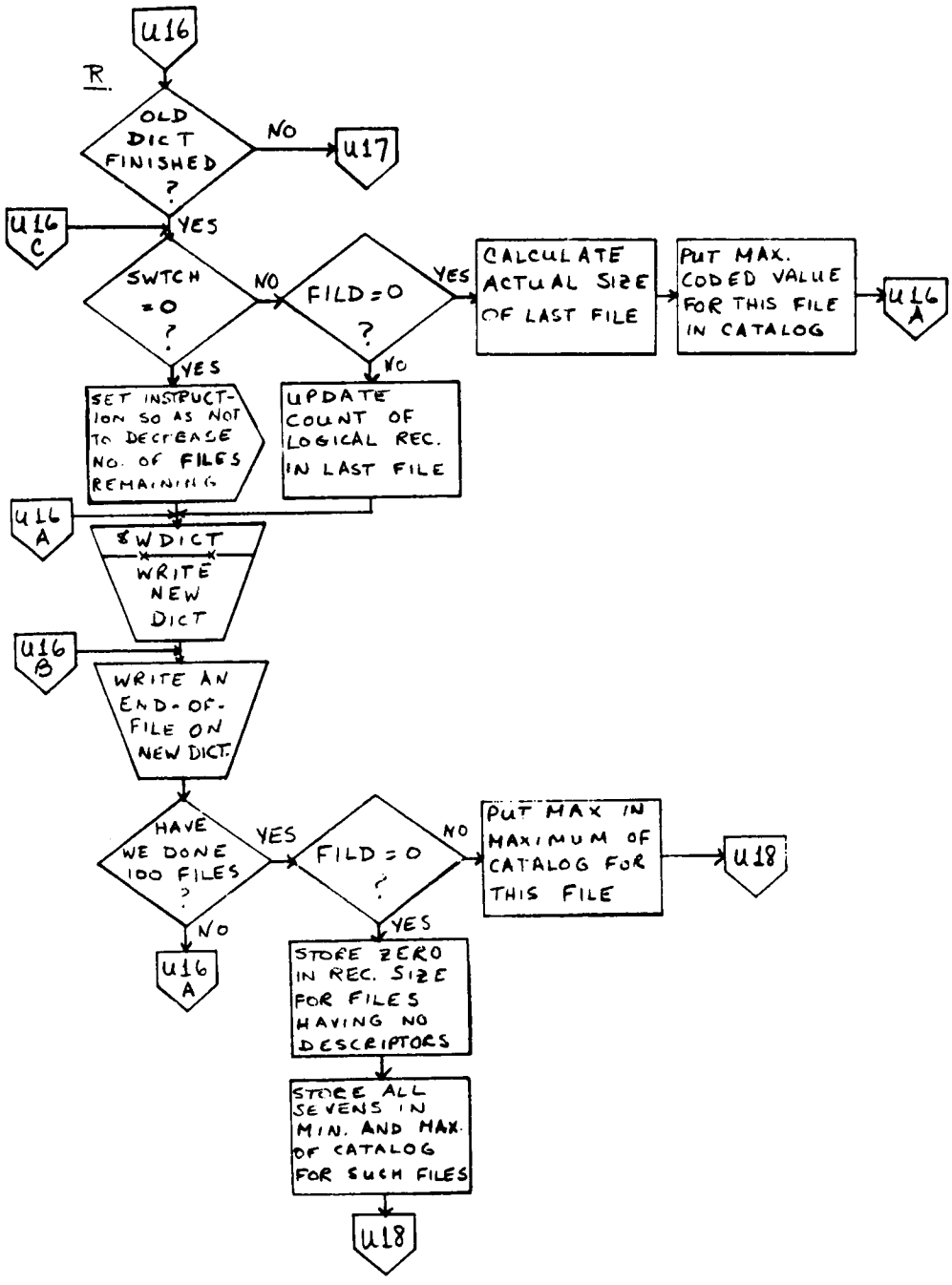

U12
IS THERE AN OLD DICT ?
NO
ADJUST THE NO. OF LOG. RECORDS IN THIS FILE
YES
STORE MAX FOR THIS FILE IN CATALOG
U16A
IS OLD DICT. FINISHED ?
YES
NO
NEW DICT. BUFFER FULL ?
NO
U17
YES
$WDICT
WRITE NEW DICT.
SET NEW DICT COUNTER = 0
U17

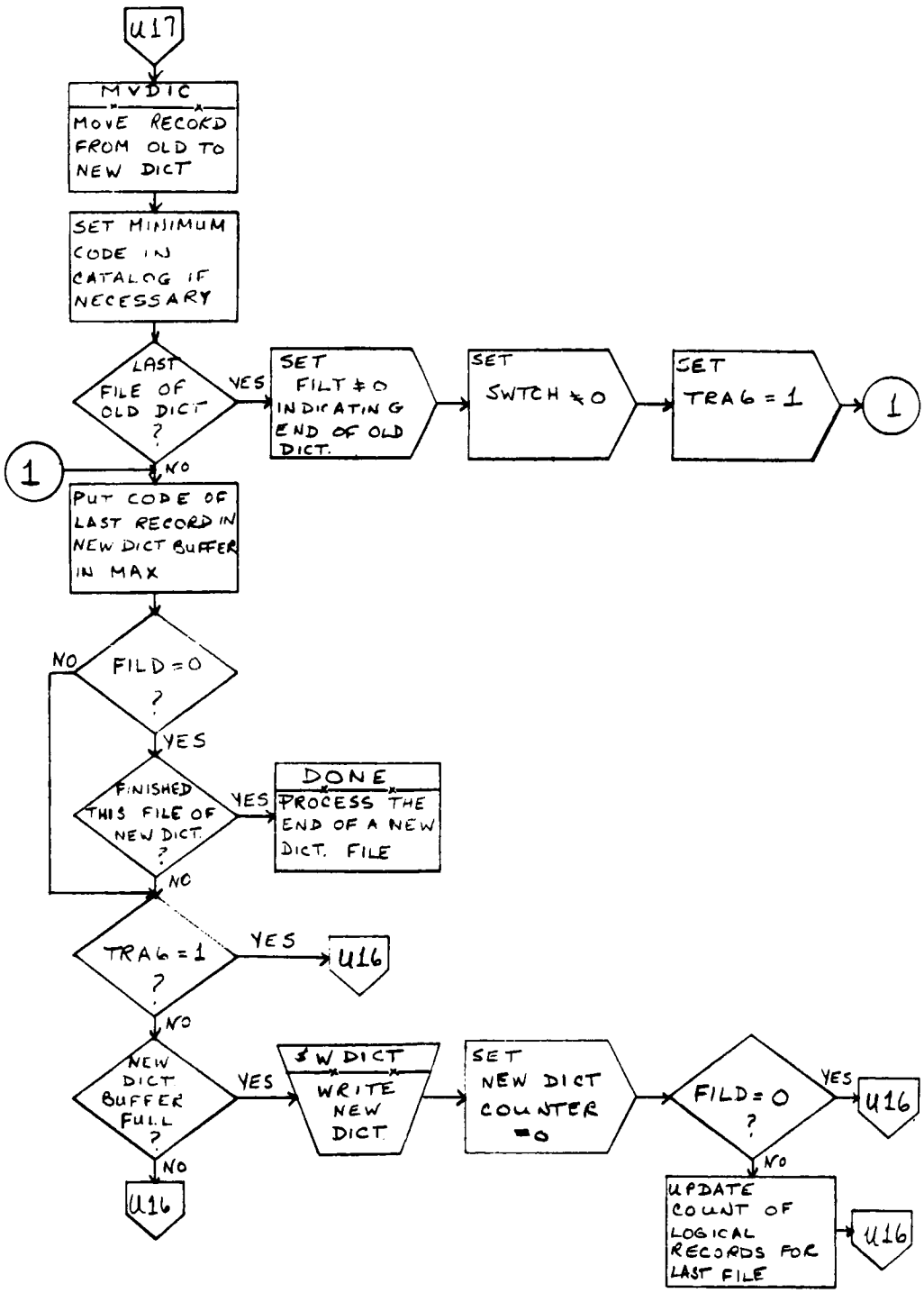

U12 A
SET UP MESSAGE - CODES EQUAL BUT ALPHAS DIFFER
SET TYPE OF DICT. WORD IN MESSAGE
PUT THE TWO CODED VALUES OF THE DICT. WORD IN MESSAGE
$WROUT
WRITE FIRST LINE
MOVE ALPHA FROM DICT. WORD TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
SET UP MESSAGE - DESCRIPTOR NOT ADDED TO DICTIONARY
U10 C

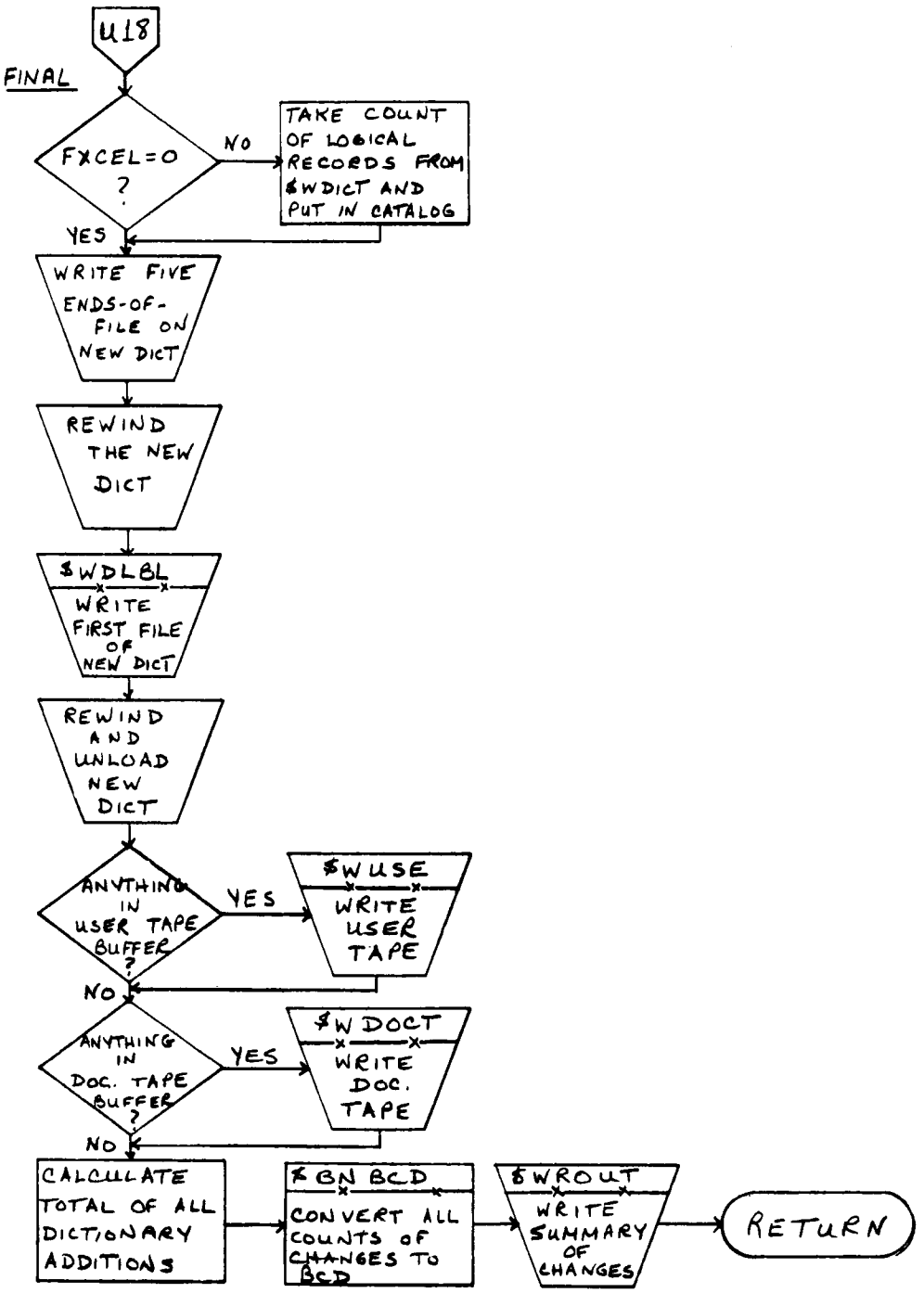

U13
DICDL
IS THERE AN OLD DICT ?
YES
MATCH ON OLD DICT ?
NO
MATCH WITH PREVIOUS CODE ?
NO
SET UP MESSAGE - DESCRIPTOR NOT ON DICTIONARY
1
SET TRA3 = 3 TRA4 = 2
U10 C
U13 C
RESET TRA3 = 2 TRA4 = 1
U13 A
ADD 1 TO BUFFER COUNTER FOR TAPE A
U13 B
FINISHED TAPE A BUFFER ?
NO
U2
YES
$READA
READ TAPE A
ALL OF TAPE A FINISHED ?
YES
U14
NO
SET TAPE A COUNTER = 0
U2
YES
SET MATCH SWITCHES = 0
UPDATE COUNT OF DICTIONARY DELETES
DELETE RECORD FROM DICTIONARY
SET UP MESSAGE - DESCRIPTOR DELETED
1
YES
SET UP MESSAGE - DELETE NOT ALLOWED IN CREATE RUN
1
NO
MATCH WITH PREVIOUS CODE ?
YES
SET UP MESSAGE - DELETE NOT ALLOWED IN CREATE RUN
U10 C
NO
SET MATCH SWITCHES = 0

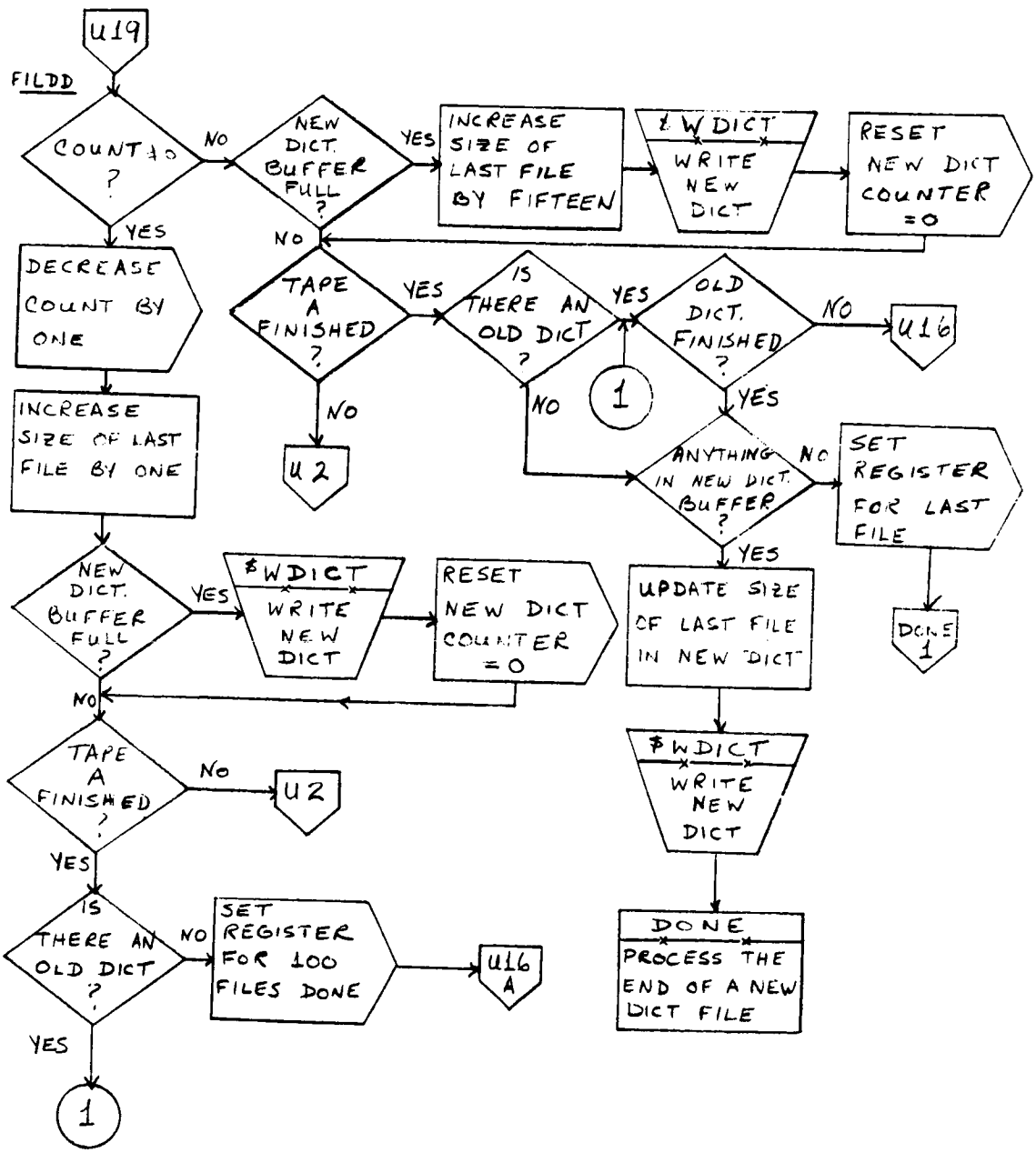

u14
DELOT
IS THERE AN OLD DICT. ?
NO
YES
OLD DICT. FINISHED ?
NO
u17
YES
LAST FILE OF NEW DICT. ?
YES
PUT ALL SEVENS IN MAX. FOR THIS FILE
SET FXCEL ≠0
$W DICT
WRITE NEW DICT.
WRITE END-OF-FILE ON NEW DICT
u18
NO
IS ANYTHING IN CURRENT FILE ?
YES
SET SWTCH ≠0
u16 C
NO
SET FILE'S SIZE = 0 AND MIN AND MAX ALL SEVENS
u16 A

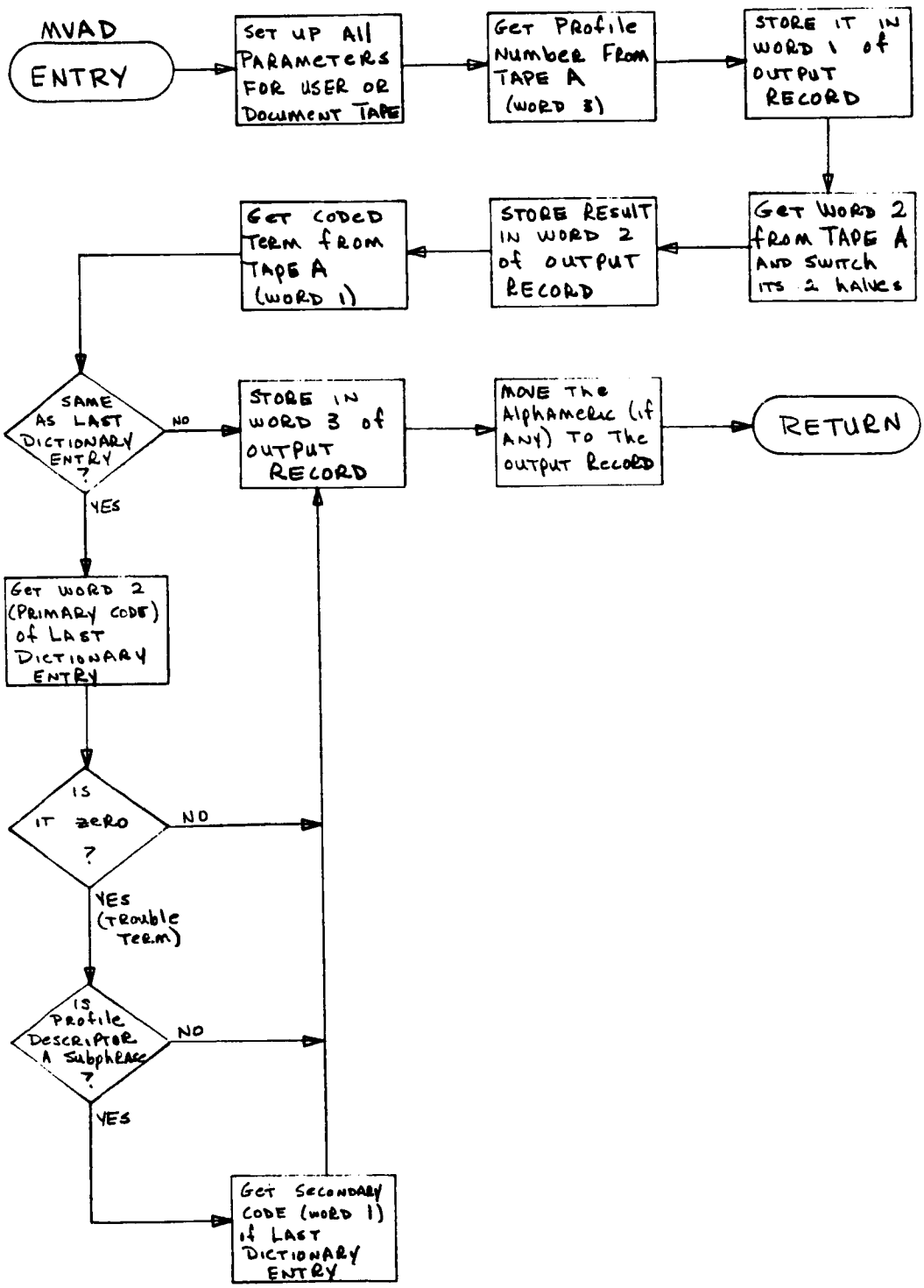

U15
DICTT
MATCH ON OLD DICT. ?
YES
NO
SET MATCH SWITCHES = 0
SET WORD2 IN NEW DICT = 0
SET UP MESSAGE - DESCRIPTOR CHANGED TO TROUBLE TERM
U1C
MATCH WITH PREVIOUS CODE ?
YES
NO
UPDATE COUNT OF DICTIONARY ADDS
FORM
SET UP RECORD IN NEW DICT BUFFER
SET WORD2 IN NEW DICT = 0
SET UP MESSAGE - TROUBLE TERM ADDED TO DICTIONARY
U10

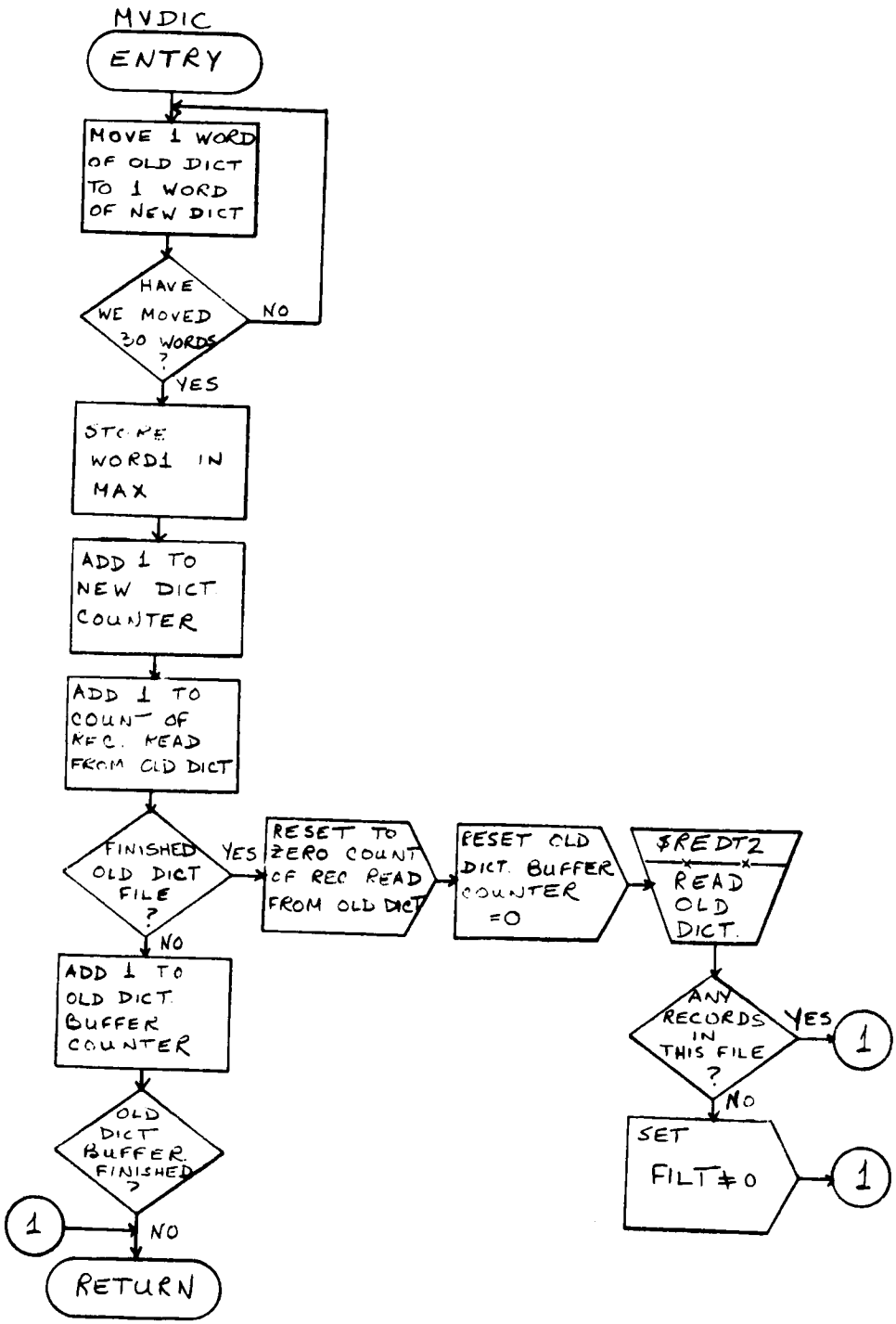

U16
R.
OLD DICT FINISHED ?
NO
U17
YES
U16 C
SWTCH =0 ?
NO
FILD=0 ?
YES
CALCULATE ACTUAL SIZE OF LAST FILE
PUT MAX. CODED VALUE FOR THIS FILE IN CATALOG
U16 A
YES
NO
SET INSTRUCTION SO AS NOT TO DECREASE NO. OF FILES REMAINING
UPDATE COUNT OF LOGICAL REC. IN LAST FILE
U16 A
$WDICT
WRITE NEW DICT
U16 B
WRITE AN END-OF-FILE ON NEW DICT.
HAVE WE DONE 100 FILES ?
YES
FILD=0 ?
NO
PUT MAX IN MAXIMUM OF CATALOG FOR THIS FILE
U18
NO
U16 A
YES
STORE ZERO IN REC. SIZE FOR FILES HAVING NO DESCRIPTORS
STORE ALL SEVENS IN MIN. AND MAX. OF CATALOG FOR SUCH FILES
U18

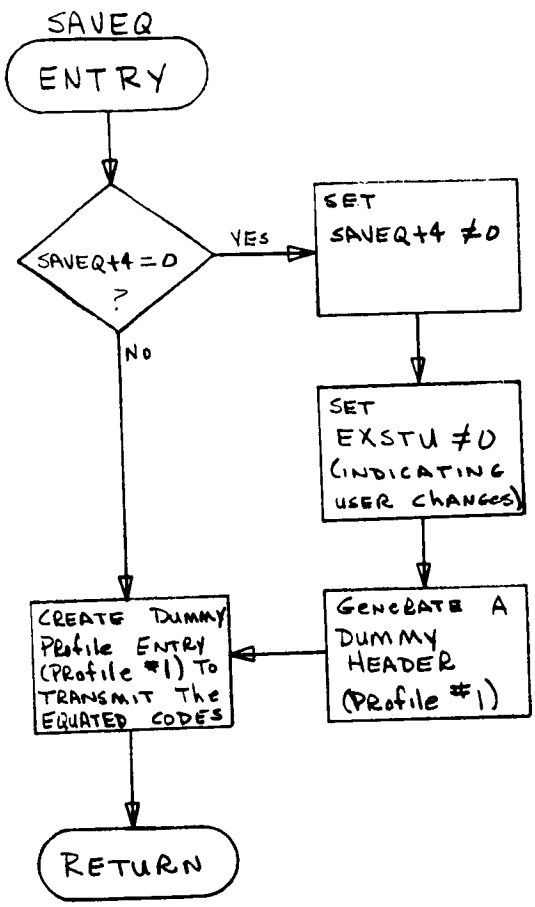

U17
MVDIC
MOVE RECORD FROM OLD TO NEW DICT
SET MINIMUM CODE IN CATALOG IF NECESSARY
LAST FILE OF OLD DICT ?
YES
SET FILT ≠ 0 INDICATING END OF OLD DICT.
SET SWTCH ≠ 0
SET TRA6 = 1
1
1
NO
PUT CODE OF LAST RECORD IN NEW DICT BUFFER IN MAX
FILD = 0 ?
NO
YES
FINISHED THIS FILE OF NEW DICT. ?
YES
DONE
PROCESS THE END OF A NEW DICT. FILE
NO
TRA6 = 1 ?
YES
U16
NO
NEW DICT BUFFER FULL ?
YES
$W DICT
WRITE NEW DICT
SET NEW DICT COUNTER =0
FILD = 0 ?
YES
U16
NO
UPDATE COUNT OF LOGICAL RECORDS FOR LAST FILE
U16
NO
U16

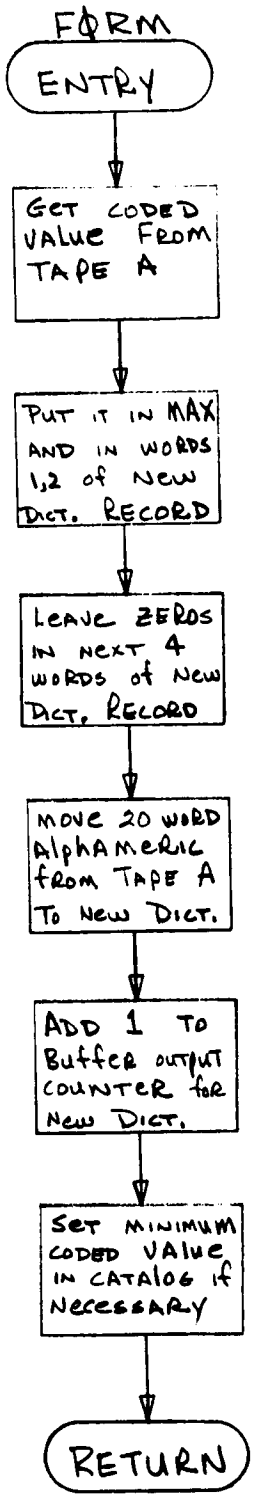

U18
FINAL
FXCEL=0 ?
NO
TAKE COUNT OF LOGICAL RECORDS FROM $WDICT AND PUT IN CATALOG
YES
WRITE FIVE ENDS-OF-FILE ON NEW DICT
REWIND THE NEW DICT
$WDLBL
WRITE FIRST FILE OF NEW DICT
REWIND AND UNLOAD NEW DICT
ANYTHING IN USER TAPE BUFFER ?
YES
$WUSE
WRITE USER TAPE
NO
ANYTHING IN DOC. TAPE BUFFER ?
YES
$WDOCT
WRITE DOC. TAPE
NO
CALCULATE TOTAL OF ALL DICTIONARY ADDITIONS
$BNBCD
CONVERT ALL COUNTS OF CHANGES TO BCD
$WROUT
WRITE SUMMARY OF CHANGES
RETURN

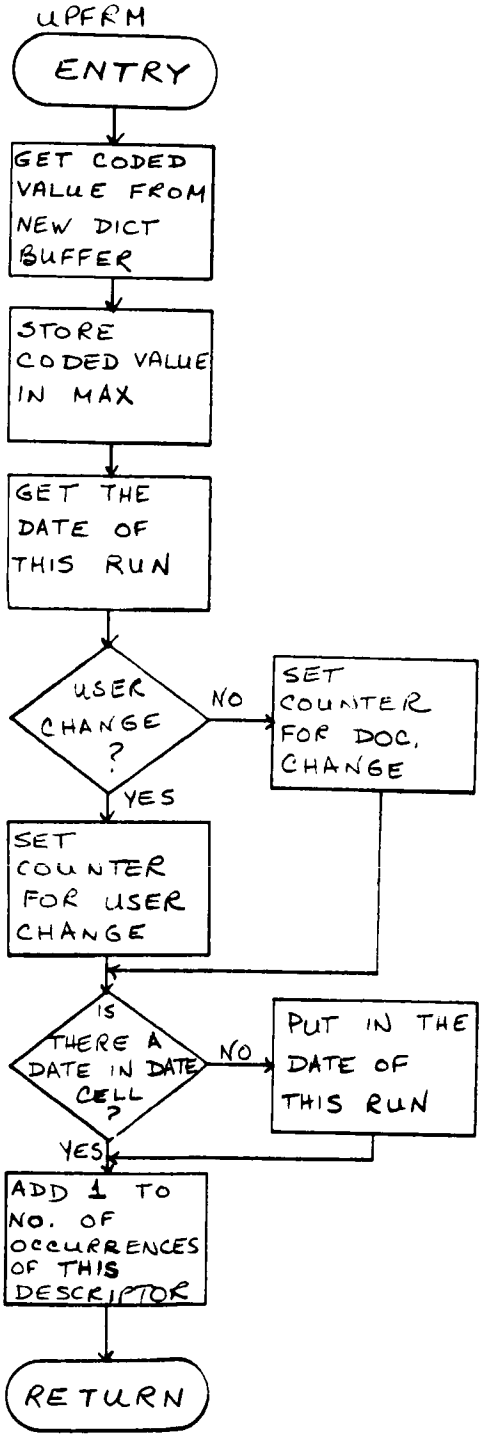

U19
FILDD
COUNT 40 ?
NO
YES
NEW DICT. BUFFER FULL ?
YES
NO
INCREASE SIZE OF LAST FILE BY FIFTEEN
$WDICT
WRITE NEW DICT
RESET NEW DICT COUNTER =0
DECREASE COUNT BY ONE
TAPE A FINISHED ?
YES
NO
U2
IS THERE AN OLD DICT ?
YES
NO
1
OLD DICT. FINISHED ?
NO
U16
YES
INCREASE SIZE OF LAST FILE BY ONE
ANYTHING IN NEW DICT. BUFFER ?
NO
YES
SET REGISTER FOR LAST FILE
DONE 1
NEW DICT. BUFFER FULL ?
YES
$WDICT
WRITE NEW DICT
RESET NEW DICT COUNTER =0
NO
UPDATE SIZE OF LAST FILE IN NEW DICT
TAPE A FINISHED ?
NO
U2
YES
$WDICT
WRITE NEW DICT
IS THERE AN OLD DICT ?
NO
SET REGISTER FOR 100 FILES DONE
U16 A
YES
1
DONE
PROCESS THE END OF A NEW DICT FILE

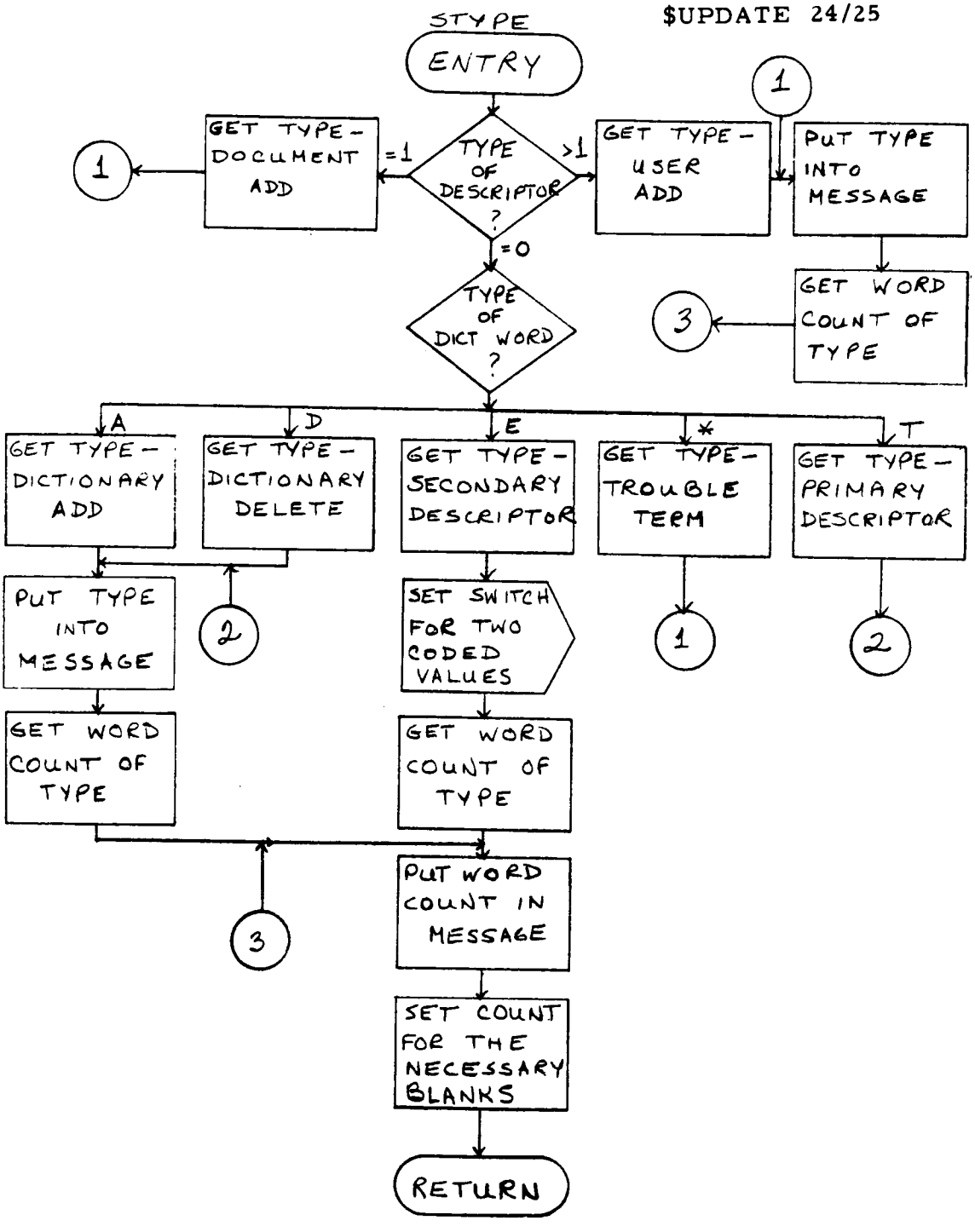

MVAD
ENTRY
Set up All Parameters for User or Document Tape
Get Profile Number From Tape A (word 8)
Store it in word 1 of output record
Get word 2 from Tape A and switch its 2 halves
Store result in word 2 of output record
Get coded term from Tape A (word 1)
Same as last Dictionary Entry ?
NO
Store in word 3 of output record
Move the Alphameric (if any) to the output Record
RETURN
YES
Get word 2 (Primary code) of last Dictionary Entry
Is it zero ?
NO
YES (Trouble Term)
Is Profile Descriptor A Subphrase ?
NO
YES
Get Secondary code (word 1) if last Dictionary Entry

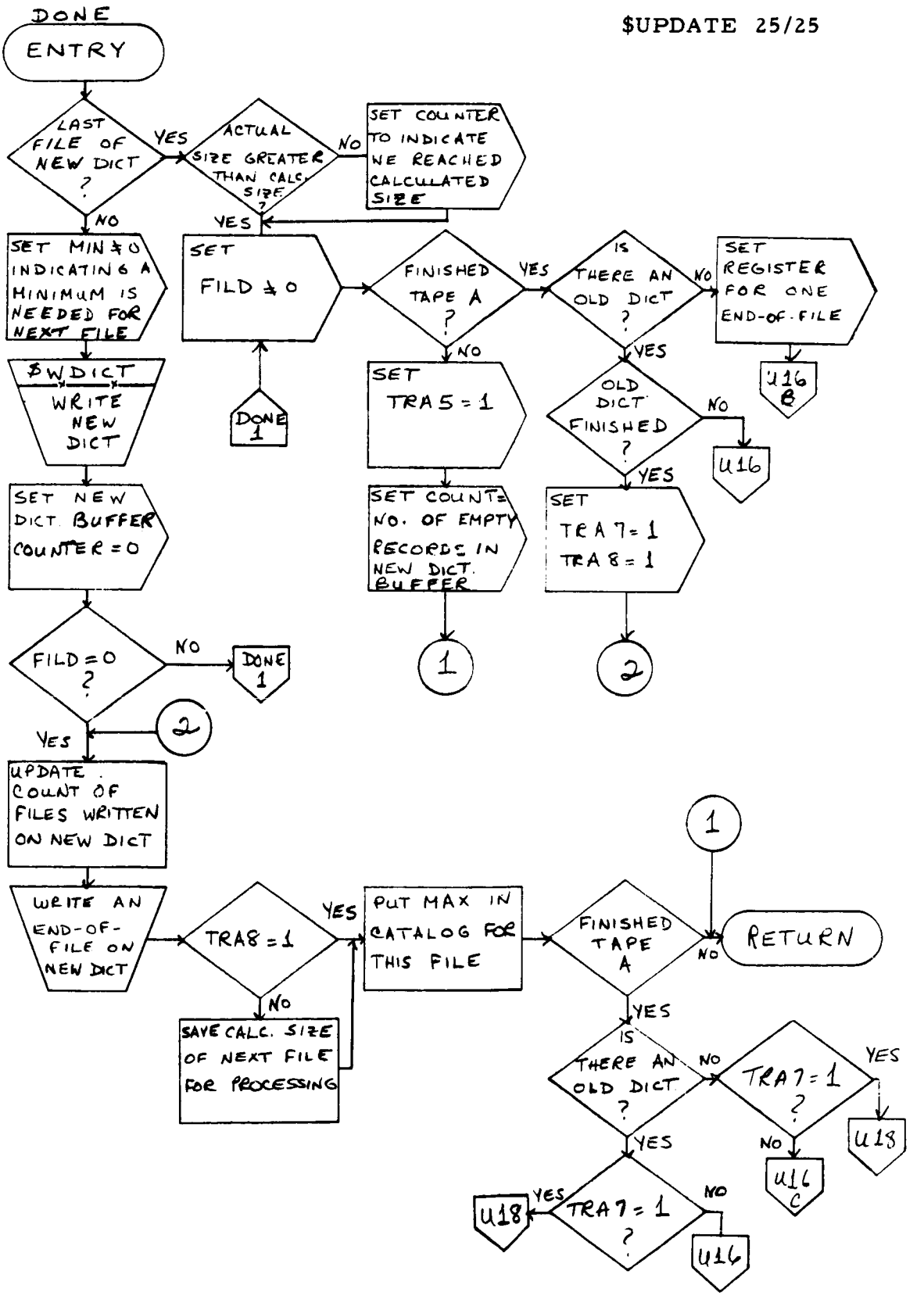

MVDIC
ENTRY
MOVE 1 WORD OF OLD DICT TO 1 WORD OF NEW DICT
HAVE WE MOVED 30 WORDS ?
NO
YES
STORE WORD1 IN MAX
ADD 1 TO NEW DICT COUNTER
ADD 1 TO COUNT OF REC. READ FROM OLD DICT
FINISHED OLD DICT FILE ?
YES
RESET TO ZERO COUNT OF REC READ FROM OLD DICT
RESET OLD DICT. BUFFER COUNTER =0
$READT2
READ OLD DICT.
ANY RECORDS IN THIS FILE ?
YES
1
NO
SET FILT≠0
1
NO
ADD 1 TO OLD DICT BUFFER COUNTER
OLD DICT BUFFER FINISHED ?
1
NO
RETURN


SAVEQ
ENTRY
SAVEQ+4 = 0 ?
YES
NO
SET SAVEQ+4 ≠0
SET EXSTU ≠0 (INDICATING USER CHANGES)
GENERATE A DUMMY HEADER (PROFILE #1)
CREATE DUMMY PROFILE ENTRY (PROFILE #1) TO TRANSMIT THE EQUATED CODES
RETURN


FORM
ENTRY
GET CODED VALUE FROM TAPE A
PUT IT IN MAX AND IN WORDS 1,2 of NEW DICT. RECORD
LEAVE ZEROS IN NEXT 4 WORDS of NEW DICT. RECORD
MOVE 20 WORD AlphAMERIC FROM TAPE A TO NEW DICT.
ADD 1 TO Buffer output COUNTER for NEW DICT.
SET MINIMUM CODED VALUE IN CATALOG if NECESSARY
RETURN


UPFRM
ENTRY
GET CODED VALUE FROM NEW DICT BUFFER
STORE CODED VALUE IN MAX
GET THE DATE OF THIS RUN
USER CHANGE ?
NO
SET COUNTER FOR DOC. CHANGE
YES
SET COUNTER FOR USER CHANGE
IS THERE A DATE IN DATE CELL ?
NO
PUT IN THE DATE OF THIS RUN
YES
ADD 1 TO NO. OF OCCURRENCES OF THIS DESCRIPTOR
RETURN


STYPE
ENTRY
TYPE OF DESCRIPTOR ?
=1
>1
=0
GET TYPE - DOCUMENT ADD
1
GET TYPE - USER ADD
1
PUT TYPE INTO MESSAGE
GET WORD COUNT OF TYPE
3
TYPE OF DICT WORD ?
A
D
E
*
T
GET TYPE - DICTIONARY ADD
GET TYPE - DICTIONARY DELETE
GET TYPE - SECONDARY DESCRIPTOR
GET TYPE - TROUBLE TERM
GET TYPE - PRIMARY DESCRIPTOR
PUT TYPE INTO MESSAGE
2
SET SWITCH FOR TWO CODED VALUES
1
2
GET WORD COUNT OF TYPE
GET WORD COUNT OF TYPE
3
PUT WORD COUNT IN MESSAGE
SET COUNT FOR THE NECESSARY BLANKS
RETURN


DONE
ENTRY
LAST FILE OF NEW DICT ?
YES
ACTUAL SIZE GREATER THAN CALC. SIZE ?
NO
SET COUNTER TO INDICATE NE REACHED CALCULATED SIZE
NO
YES
SET MIN ≠ 0 INDICATING A MINIMUM IS NEEDED FOR NEXT FILE
SET FILD ≠ 0
FINISHED TAPE A ?
YES
IS THERE AN OLD DICT ?
NO
SET REGISTER FOR ONE END-OF-FILE
U16 B
$W DICT WRITE NEW DICT
DONE 1
NO
SET TRA 5 = 1
YES
OLD DICT FINISHED ?
NO
U16
SET NEW DICT BUFFER COUNTER = 0
SET NO. OF EMPTY RECORDS IN NEW DICT. BUFFER
YES
SET TRA 7 = 1 TRA 8 = 1
FILD = 0 ?
NO
DONE 1
1
2
2
YES
UPDATE COUNT OF FILES WRITTEN ON NEW DICT
WRITE AN END-OF-FILE ON NEW DICT
TRA8 = 1
YES
PUT MAX IN CATALOG FOR THIS FILE
FINISHED TAPE A
1
RETURN
NO
SAVE CALC. SIZE OF NEXT FILE FOR PROCESSING
NO
YES
IS THERE AN OLD DICT ?
NO
TRA 7 = 1 ?
YES
U18
NO
U16 C
YES
U18
TRA 7 = 1 ?
YES
NO
U16

Subroutine UPDUS

The building and updating of the two user tapes (coded and historical) is done in this subroutine.

There are five sections within UPDUS that read or write all binary tapes. All are initialized upon their first entry and take appropriate action when encountering redundancy, EOF or EOT. These sections are:

READC	Read sorted tape C
USRID	Read old user tape
UHID	Read old historical profiles tape
WRUID	Write new user tape
WRHID	Write new historical profiles tape.

The FORTRAN II I/O Package is used whenever it is desired to write on the system output tape. This is accomplished with a TSX (TAPE), 4 and two calling sequence words.

First in UPDUS, all counters are set to zero. Flags are then set to show whether or not there is an IP user tape, IP historical profile tape and whether or not on an initial run a historical profile tape is to be generated.

If there is an IP user tape, its identification record is checked against the IP user control card. If there is not a match, a message will be printed on line and a dump will be taken. The same is done if there is an IP historical profiles tape.

The new identification records, with the date of the run, are written on the new user tape (and historical tape, if there is one).

The two types of runs, build and update, will be discussed separately.

Build. All input for building user tapes comes from tape C. The first descriptor for any user must be a header. If there is no header, the user will not be processed and a message will be written on the system output tape. Each add descriptor is checked to see if it should be added to the user profile. When all the adds have been processed, the percent that have been accepted will be compared to the required percentage (MINU) specified on one of the control cards. If the accept percentage is greater than or equal to MINU, the user profile will be written on the binary tape. If not, it will be rejected. If the profile has been accepted, it will be written on the historical profile tape if that tape is specified. Descriptor delete cards will not be processed on a build computer run.

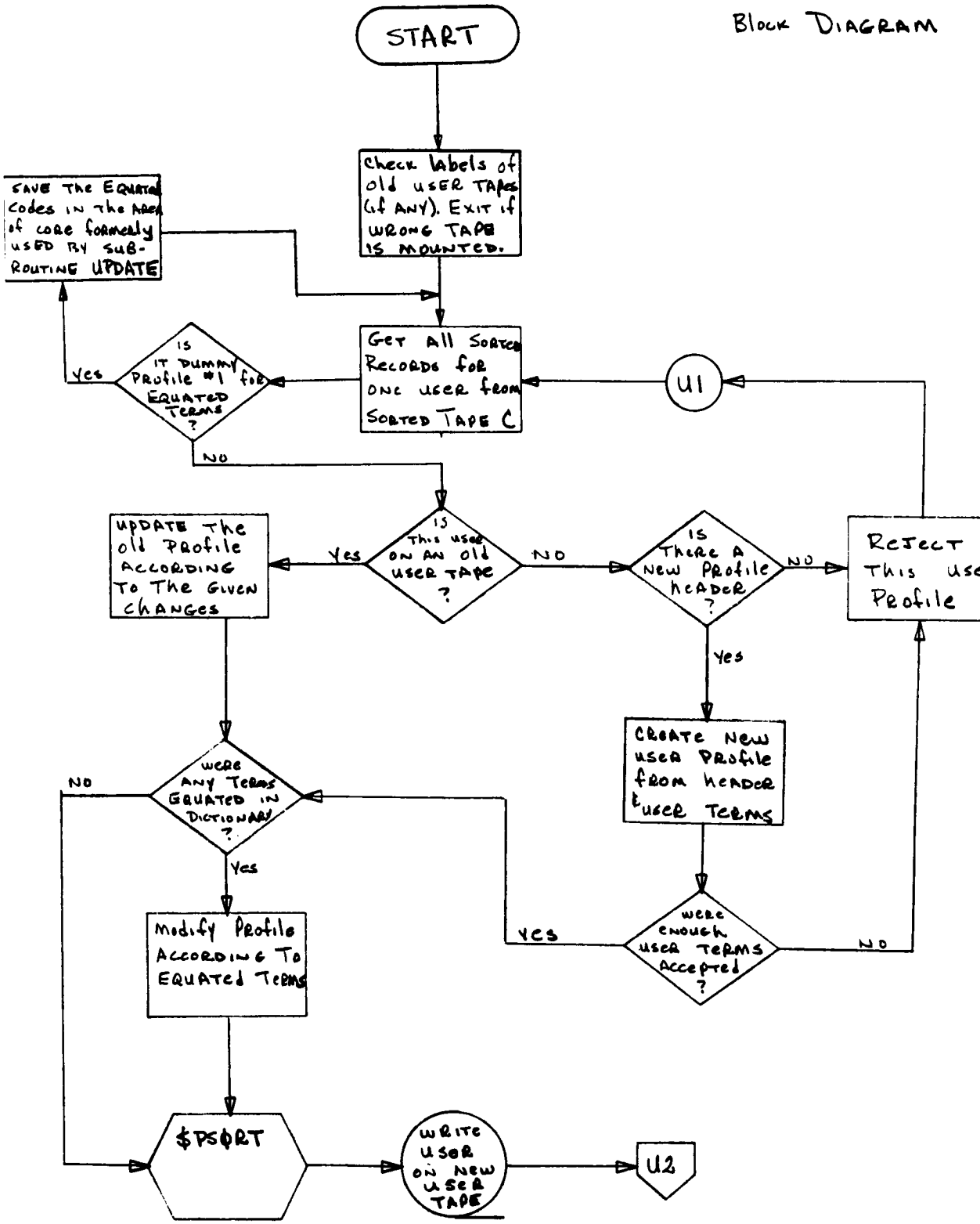
Update. On an update run there are two, or possibly three, input tapes. The two required are tape C and the IP user tape. If this run is with historical profiles there will also be an IP historical profile tape.

Users on the old tape(s) that do not appear on tape C are simply copied onto the new binary tape(s). When a user on the old tape also appears on tape C, the descriptors on tape C will be added to or deleted from the old user record. The appearance of a header card for an existing user will be interpreted as a new header and will be substituted for the original header. No comparison is made with MINU when updating an old user profile, but a new user, introduced in an update run, will be handled as in a build run.

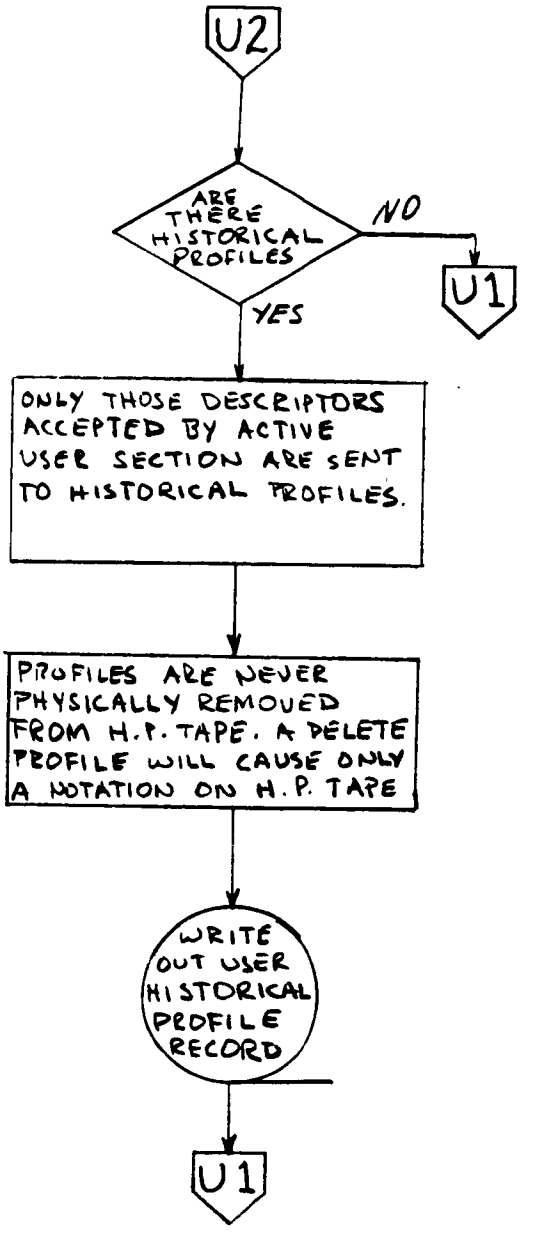
Any changes to user profiles necessitated by vocabulary equate changes (establishing synonyms) were transmitted earlier by subroutine UPDATE in a dummy user profile (profile #1). These are gathered and stored in the area of core previously used by subroutine UPDATE. Each profile written onto the new user tape is compared with the newly created synonym relationships and is modified as required.

$UPDUS 1/2

BLOCK DIAGRAM

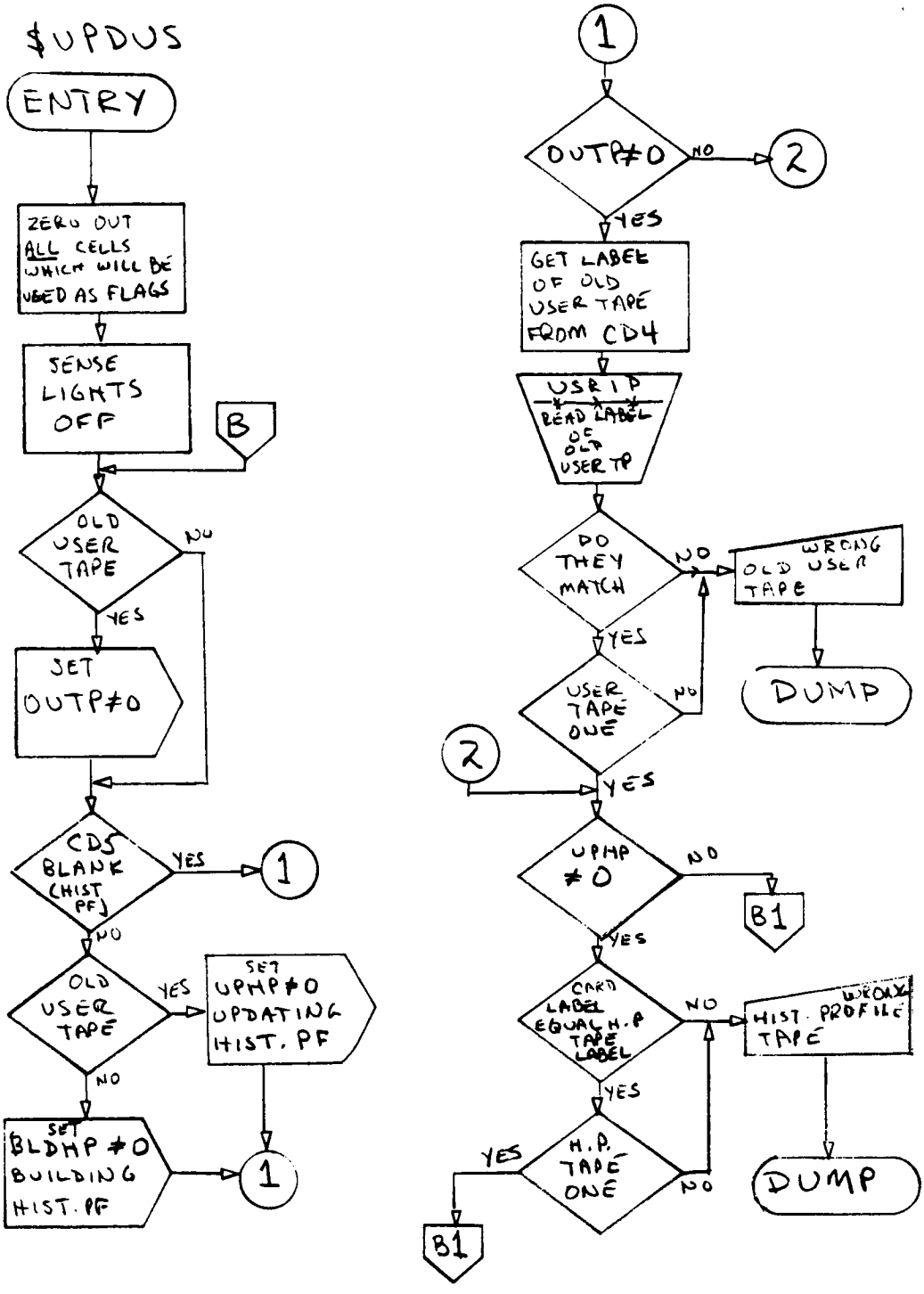


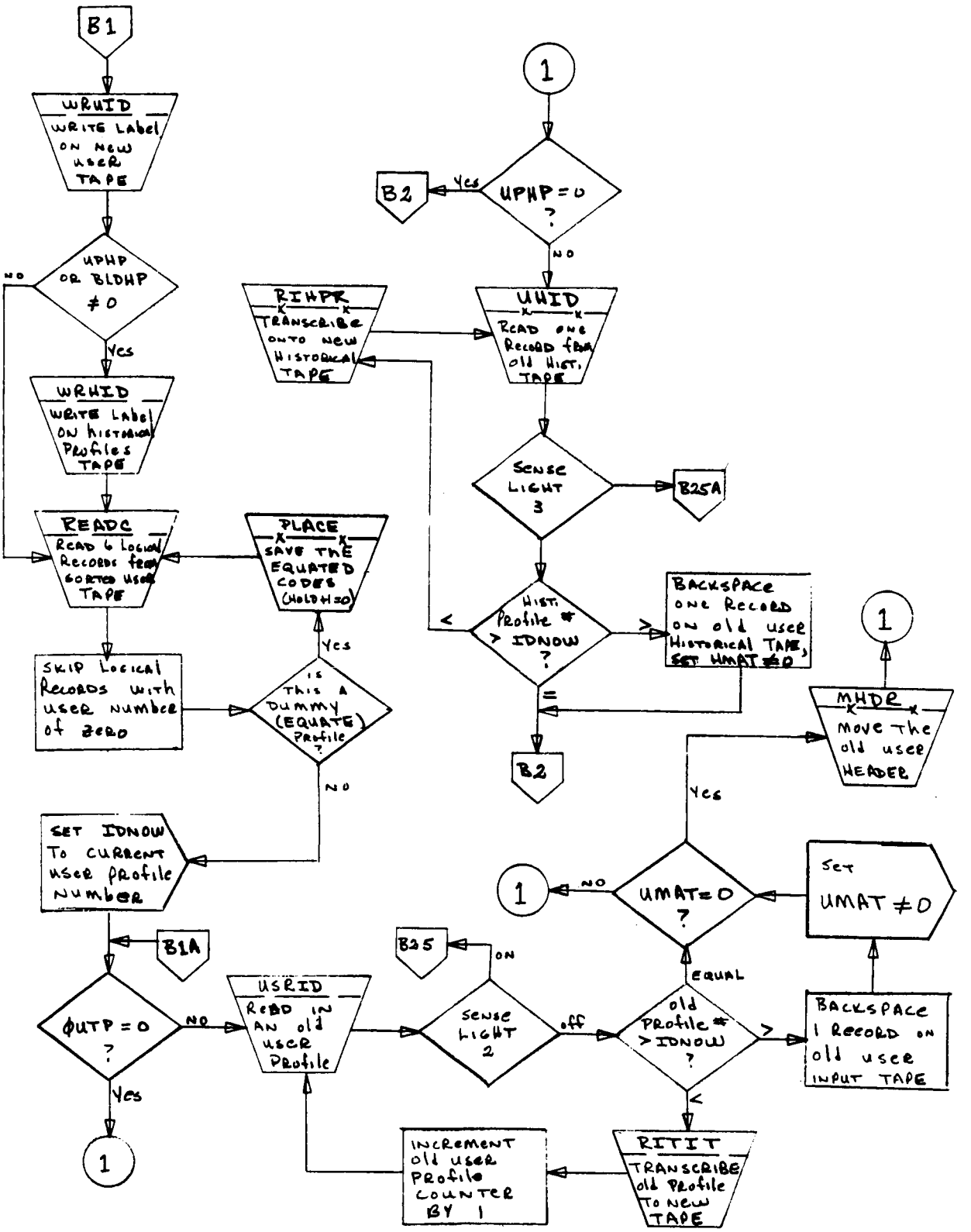
Block Diagram

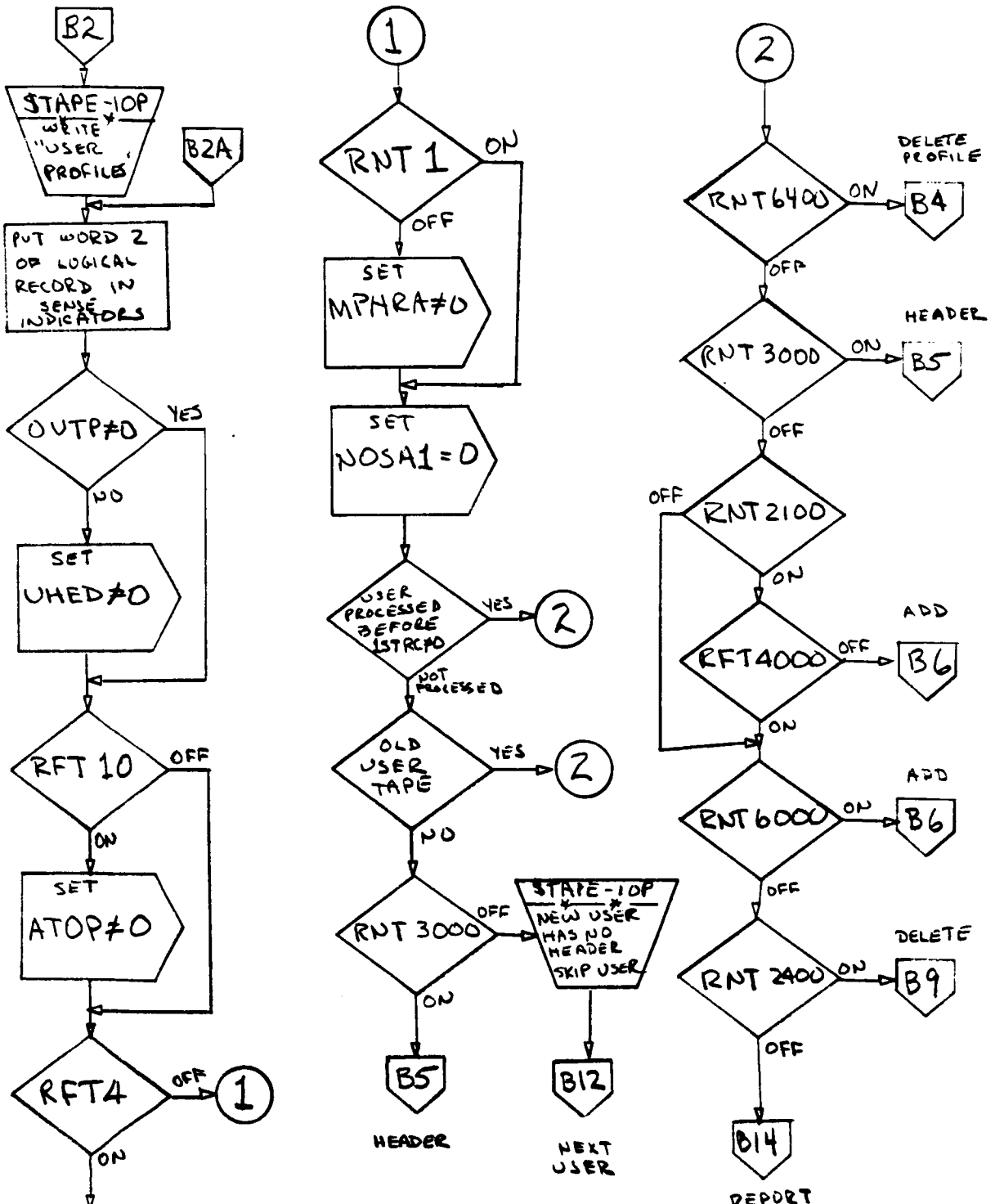


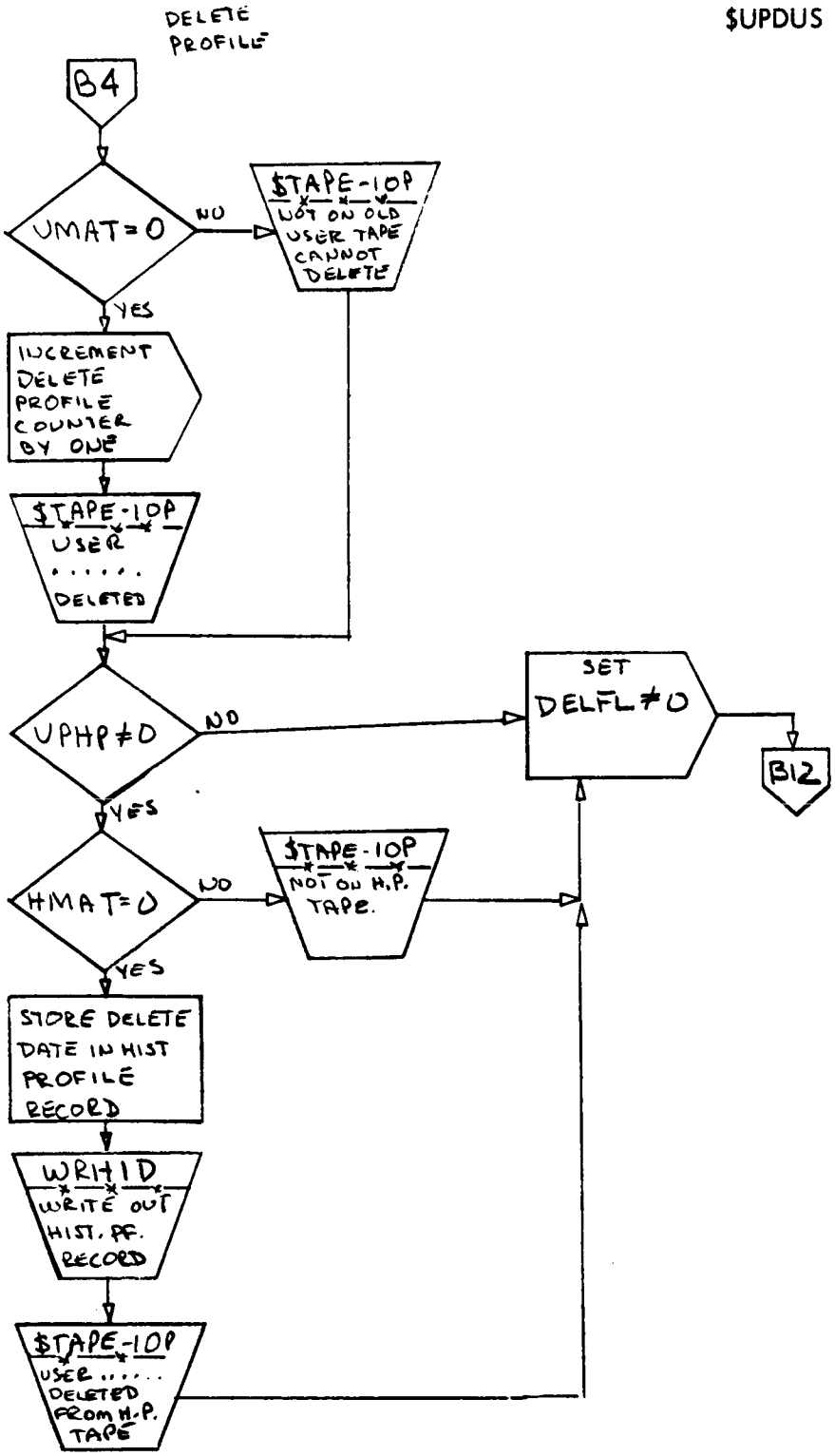
EVERY DESCRIPTOR, WHETHER ACCEPTED OR REJECTED IS SO NOTED ON THE BCD OUTPUT TAPE. THE SAME IS DONE FOR EACH PROFILE.

ON AN UPDATE RUN, ALL USERS WHO ARE NOT UPDATED ARE SIMPLY COPIED FROM OLD TAPE TO NEW TAPE.

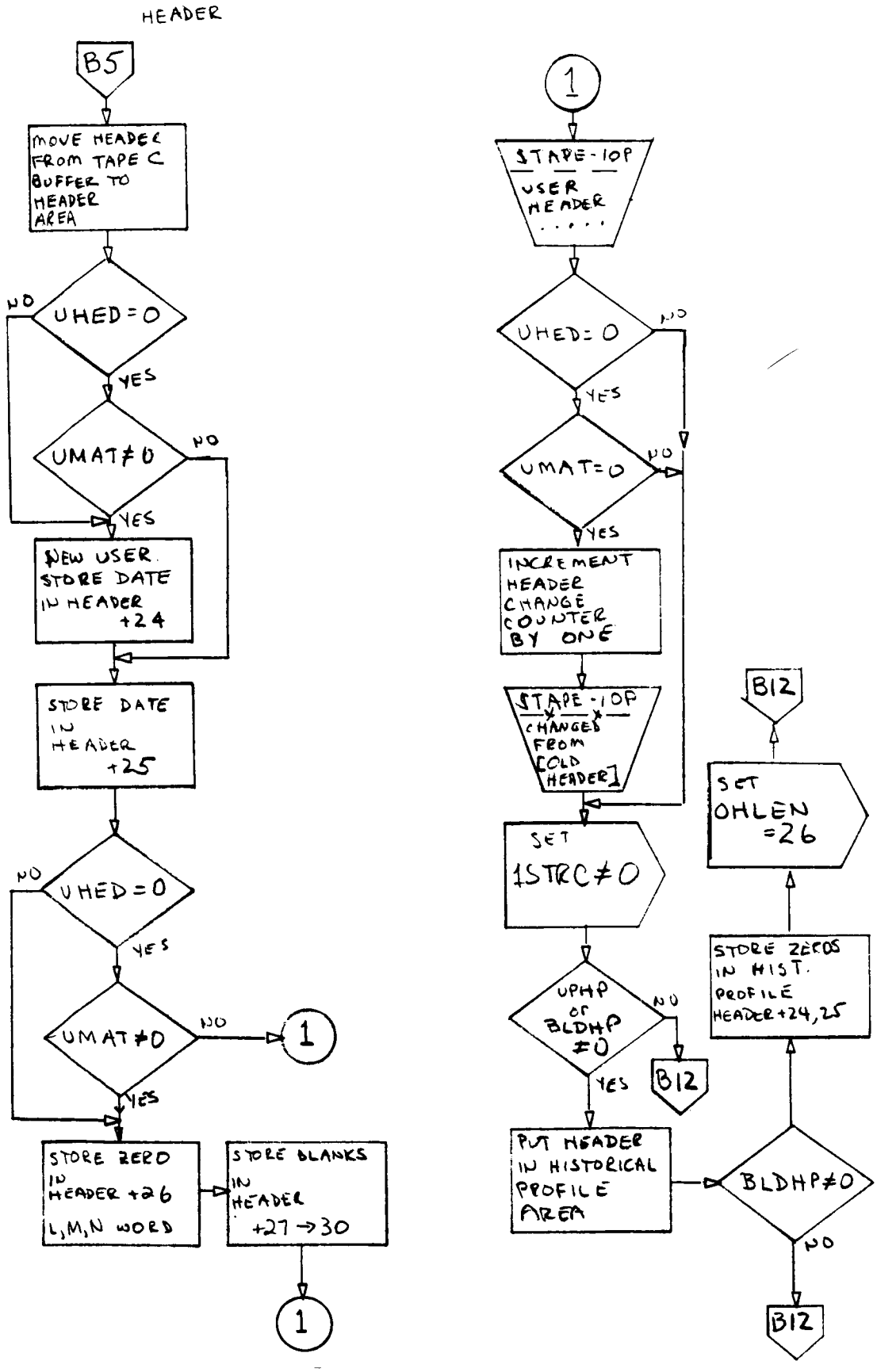

$UPDUS
ENTRY
ZERO OUT ALL CELLS WHICH WILL BE USED AS FLAGS
SENSE LIGHTS OFF
B
OLD USER TAPE
NO
YES
SET OUTP≠0
CD5 BLANK (HIST PF)
YES
1
NO
OLD USER TAPE
YES
SET UPMP≠0 UPDATING HIST. PF
NO
SET BLDMP≠0 BUILDING HIST. PF
1
1
OUTP≠0
NO
2
YES
GET LABEL OF OLD USER TAPE FROM CD4
USR I P
READ LABEL OF OLD USER TP
DO THEY MATCH
NO
WRONG OLD USER TAPE
DUMP
YES
USER TAPE ONE
NO
2
YES
UPMP ≠ 0
NO
B1
YES
CARD LABEL EQUAL H.P TAPE LABEL
NO
WRONG HIST. PROFILE TAPE
DUMP
YES
H.P. TAPE ONE
YES
B1
NO

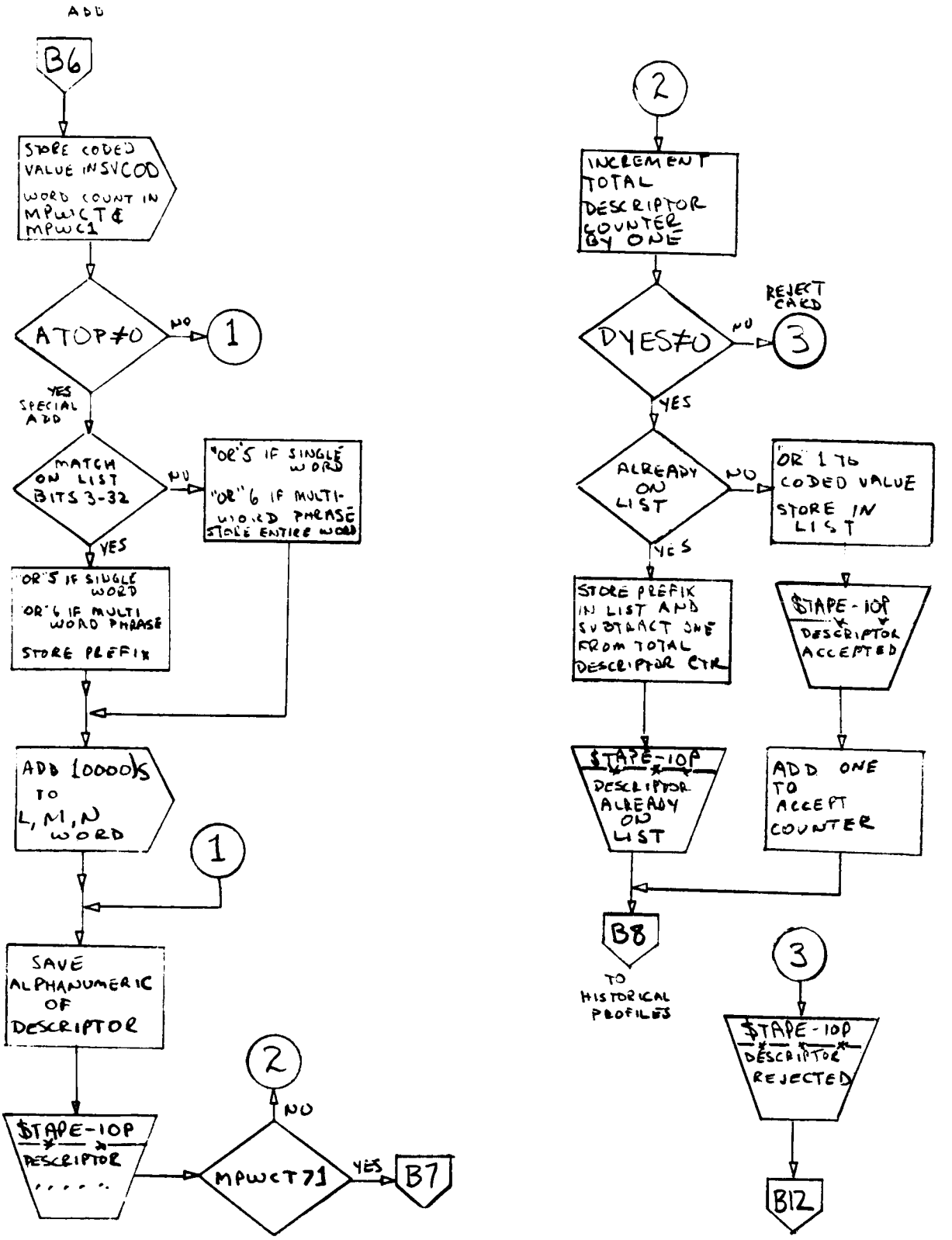

B1
WRUID
WRITE Label ON NEW USER TAPE
UPHP OR BLDHP ≠ 0
NO
Yes
WRHID
WRITE Label ON HISTORICAL Profiles TAPE
READC
READ 6 Logical Records from SORTED USER TAPE
SKIP Logical Records with user number of zero
IS THIS A DUMMY (EQUATE) Profile ?
Yes
NO
PLACE
SAVE THE EQUATED CODES (HOLDH=0)
SET IDNOW To CURRENT USER Profile Number
B1A
ΦUTP = 0 ?
NO
Yes
1
USRID
READ IN AN old USER Profile
B25
ON
SENSE LIGHT 2
off
INCREMENT old user Profile COUNTER BY 1
old Profile # > IDNOW ?
EQUAL
<
>
RITIT
TRANSCRIBE old Profile To NEW TAPE
BACKSPACE 1 RECORD ON old user INPUT TAPE
1
UMAT=0 ?
NO
Yes
Set UMAT ≠ 0
MHDR
Move the old user HEADER
1
1
UPHP = 0 ?
Yes
B2
NO
UHID
READ ONE RECORD FROM old HIST. TAPE
RIHPR
TRANSCRIBE ONTO NEW HISTORICAL TAPE
SENSE LIGHT 3
B25A
Hist. Profile # > IDNOW ?
<
>
=
BACKSPACE ONE RECORD ON old user HISTORICAL TAPE, SET UMAT ≠ 0
B2

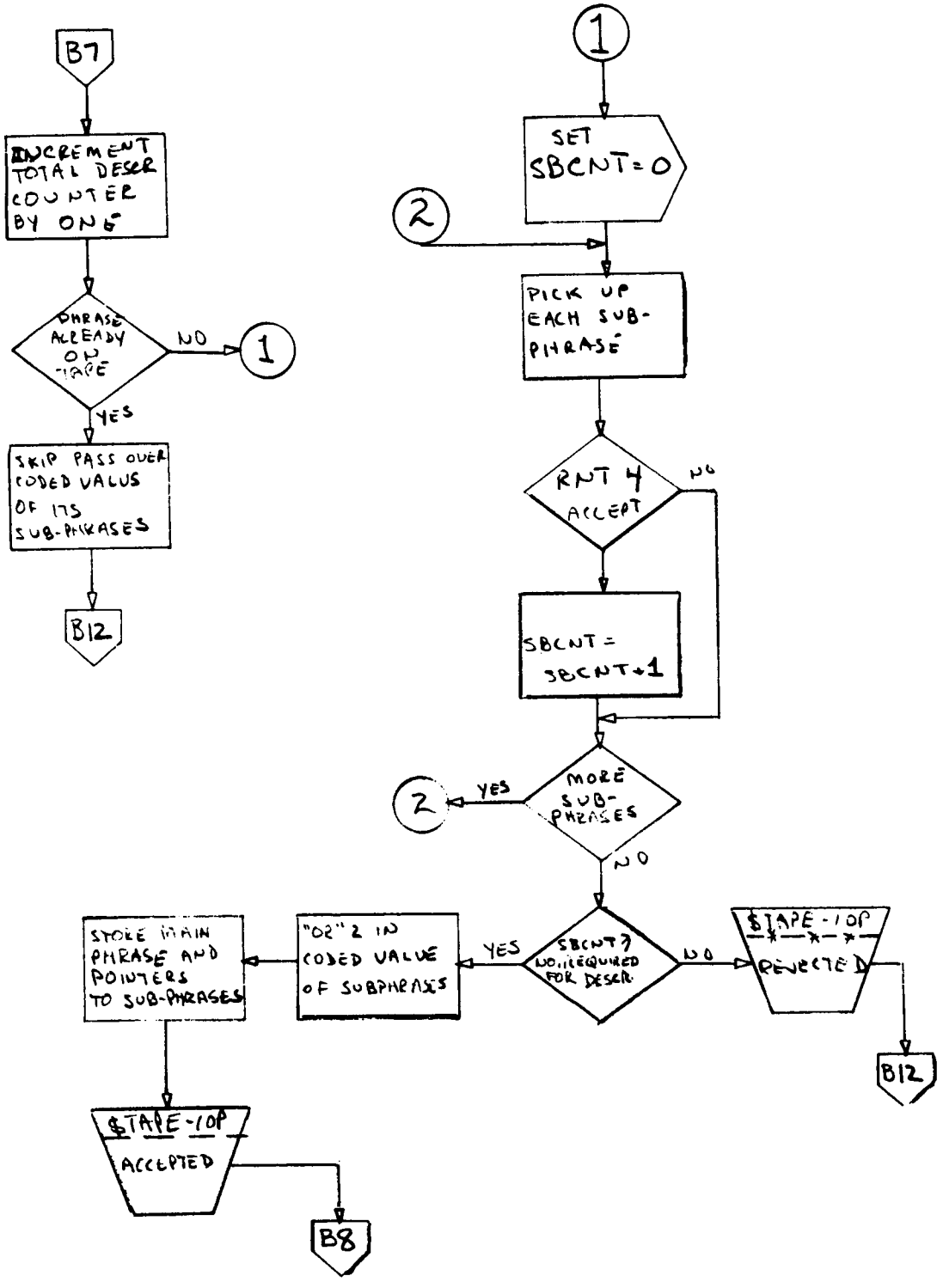

B2
$TAPE-IOP
WRITE "USER PROFILES"
B2A
PUT WORD 2 OF LOGICAL RECORD IN SENSE INDICATORS
OUTP≠0
YES
NO
SET UHED≠0
RFT 10
OFF
ON
SET ATOP≠0
RFT4
OFF
1
ON
SET DYES≠0
1
RNT 1
ON
OFF
SET MPHRA≠0
SET NOSA1=0
USER PROCESSED BEFORE ISTRC=0
YES
2
NOT PROCESSED
OLD USER TAPE
YES
2
NO
RNT 3000
OFF
$TAPE-IOP
NEW USER HAS NO HEADER SKIP USER
ON
B5
HEADER
B12
NEXT USER
2
RNT6400
ON
DELETE PROFILE
B4
OFF
RNT 3000
ON
HEADER
B5
OFF
OFF
RNT 2100
ON
RFT4000
OFF
ADD
B6
ON
RNT 6000
ON
ADD
B6
OFF
RNT 2400
ON
DELETE
B9
OFF
B14
REPORT

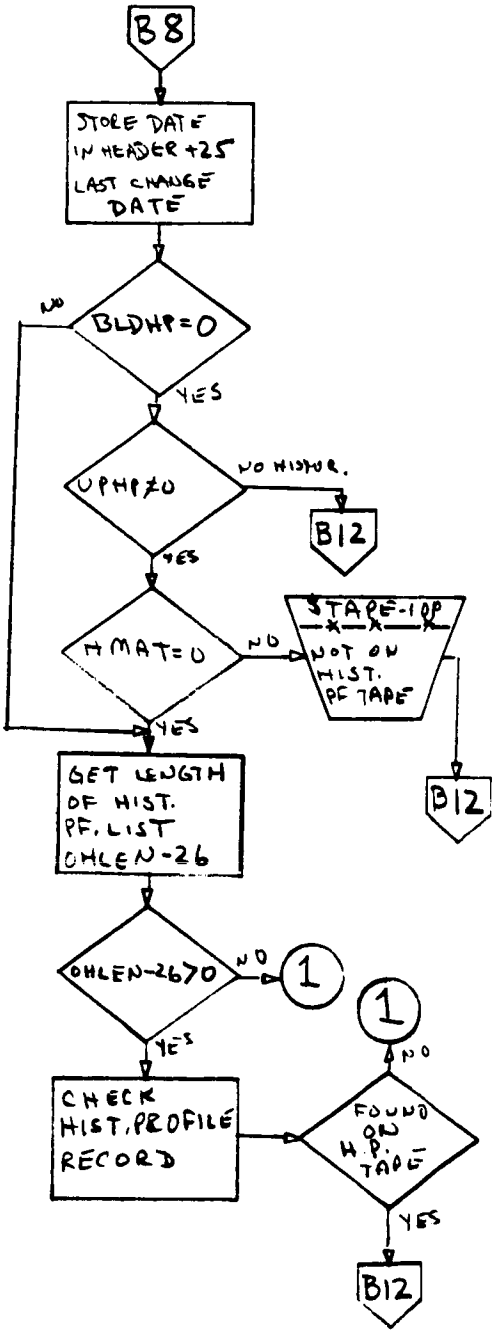

DELETE PROFILE
B4
UMAT=0
NO
$TAPE-IOP
NOT ON OLD USER TAPE CANNOT DELETE
YES
INCREMENT DELETE PROFILE COUNTER BY ONE
$TAPE-IOP
USER ..... DELETED
UPHP≠0
NO
SET DELFL≠0
B12
YES
HMAT=0
NO
$TAPE-IOP
NOT ON H.P. TAPE.
YES
STORE DELETE DATE IN HIST PROFILE RECORD
WRITID
WRITE OUT HIST. PF. RECORD
$TAPE-IOP
USER ..... DELETED FROM H.P. TAPE

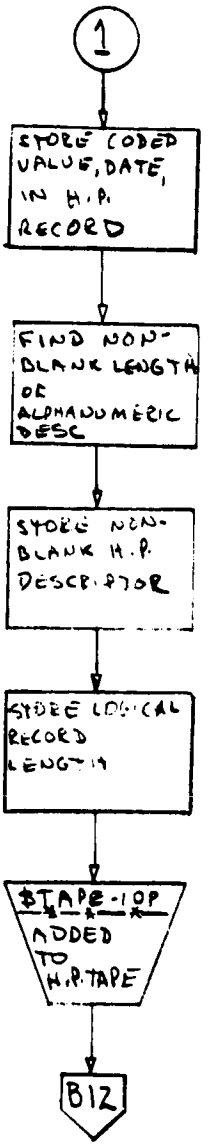
HEADER

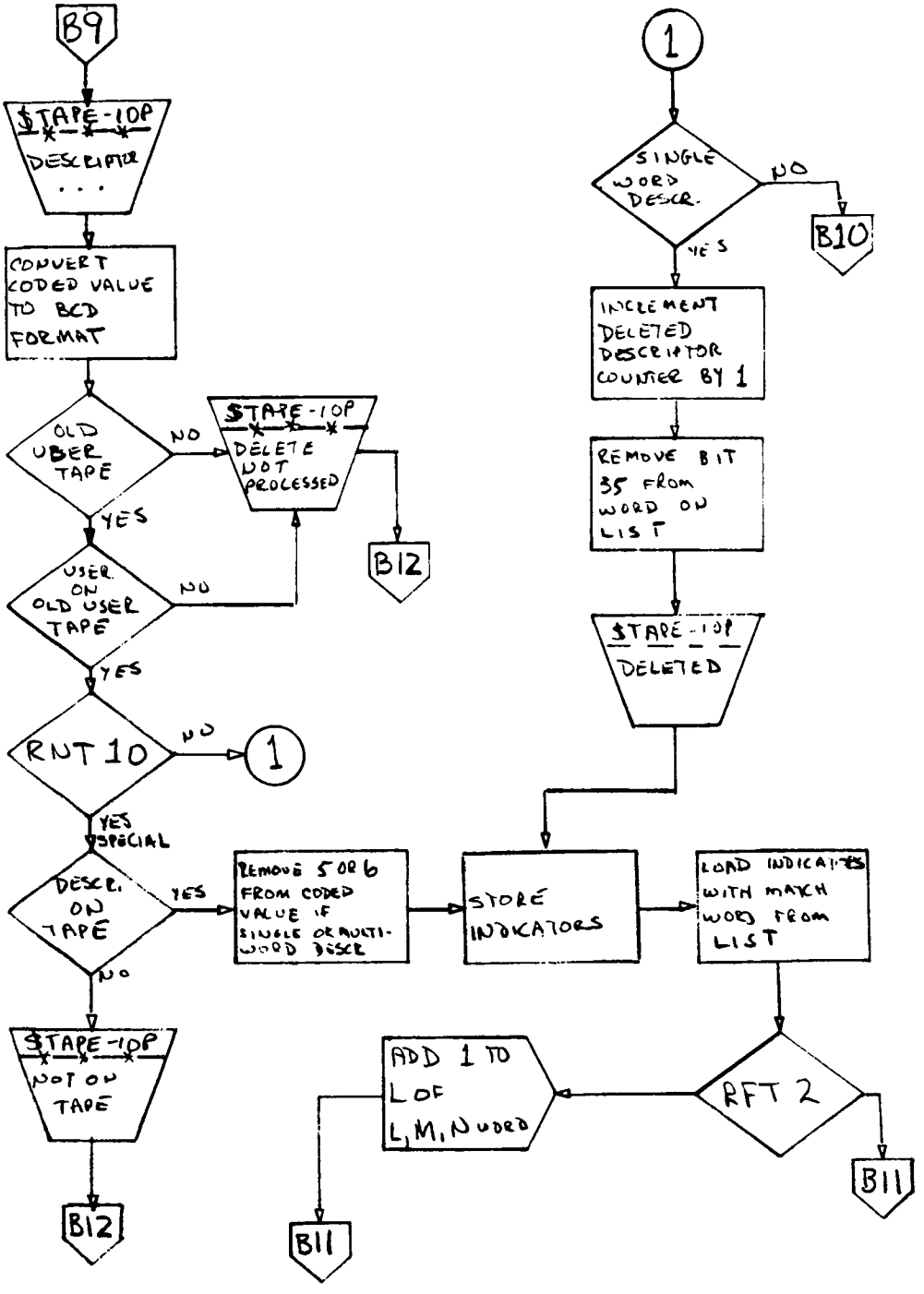

B5
MOVE HEADER FROM TAPE C BUFFER TO HEADER AREA
UHED=0
NO
YES
UMAT≠0
NO
YES
$NEW USER. STORE DATE IN HEADER +24
STORE DATE IN HEADER +25
UHED=0
NO
YES
UMAT≠0
NO
YES
1
STORE ZERO IN HEADER +26 L,M,N WORD
STORE BLANKS IN HEADER +27→30
1
1
$TAPE-IOP USER HEADER
UHED=0
NO
YES
UMAT=0
NO
YES
INCREMENT HEADER CHANGE COUNTER BY ONE
$TAPE-IOP CHANGES FROM [OLD HEADER]
SET 1STRC≠0
UPHP or BLDHP ≠0
NO
YES
B12
PUT HEADER IN HISTORICAL PROFILE AREA
BLDHP≠0
NO
B12
STORE ZEROS IN HIST. PROFILE HEADER+24,25
SET OHLEN =26
B12

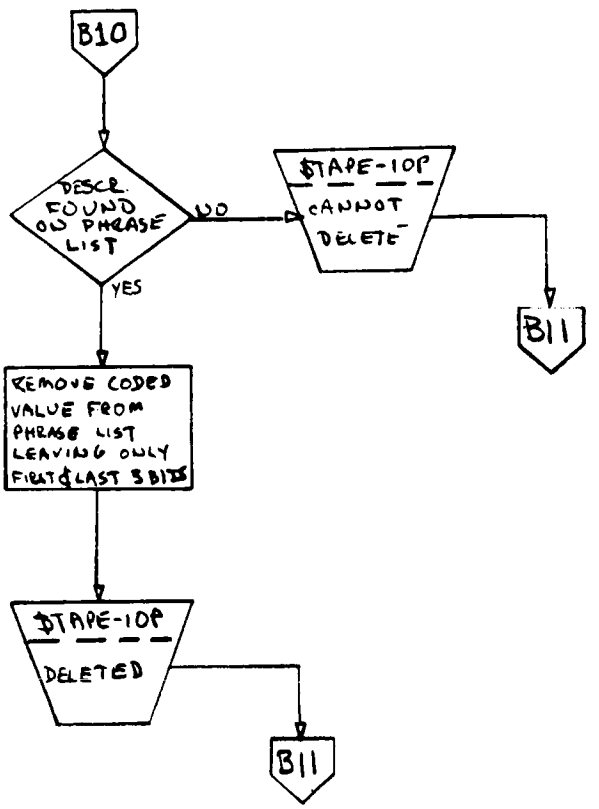

ADD
B6
STORE CODED VALUE IN SVCOD
WORD COUNT IN MPWCT & MPWC1
ATOP≠0
NO
1
YES SPECIAL ADD
MATCH ON LIST BITS 3-32
NO
"OR" 5 IF SINGLE WORD
"OR" 6 IF MULTI-WORD PHRASE
STORE ENTIRE WORD
YES
"OR" 5 IF SINGLE WORD
OR 6 IF MULTI WORD PHRASE
STORE PREFIX
ADD 100000's TO L, M, N WORD
1
SAVE ALPHANUMERIC OF DESCRIPTOR
$TAPE-IOP
DESCRIPTOR
2
NO
MPWCT 71
YES
B7
2
INCREMENT TOTAL DESCRIPTOR COUNTER BY ONE
DYES≠0
NO
REJECT CARD
3
YES
ALREADY ON LIST
NO
OR 1 TO CODED VALUE
STORE IN LIST
YES
STORE PREFIX IN LIST AND SUBTRACT ONE FROM TOTAL DESCRIPTOR CTR
$TAPE-IOP
DESCRIPTOR ACCEPTED
$TAPE-IOP
DESCRIPTOR ALREADY ON LIST
ADD ONE TO ACCEPT COUNTER
B8
TO HISTORICAL PROFILES
3
$TAPE-IOP
DESCRIPTOR REJECTED
B12

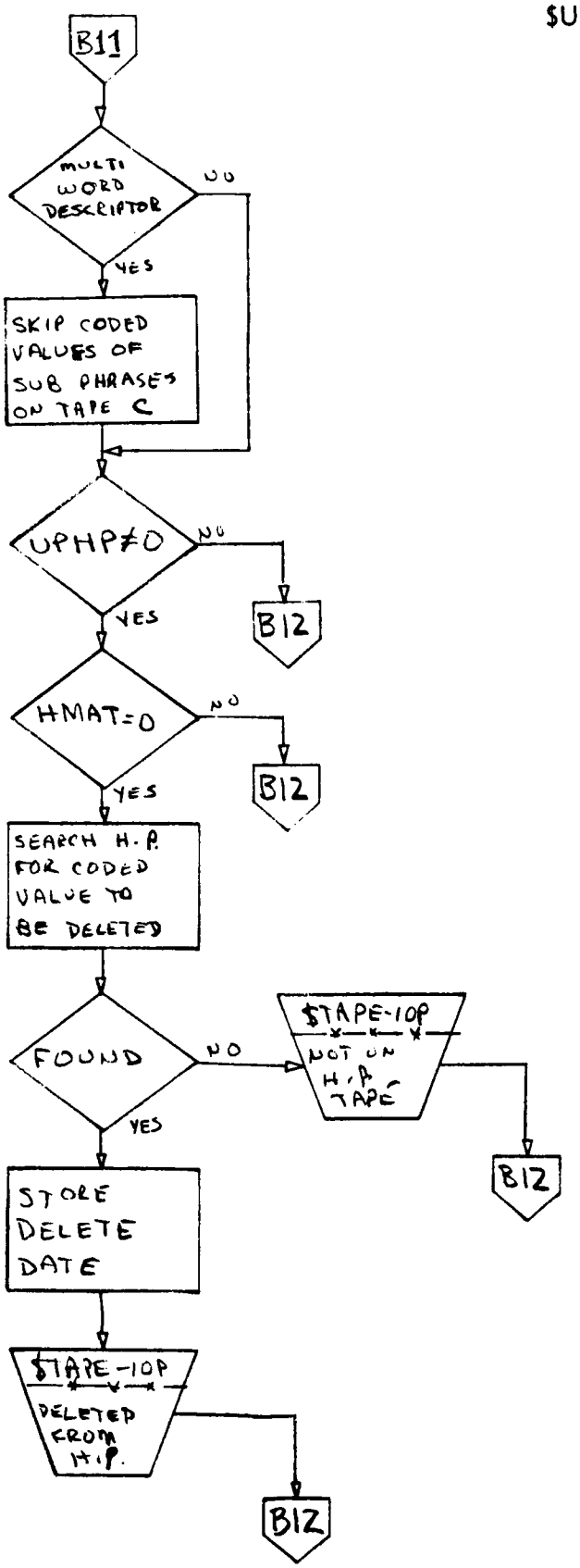

B7
INCREMENT TOTAL DESCR COUNTER BY ONE
PHRASE ALREADY ON TAPE
NO
1
YES
SKIP PASS OVER CODED VALUS OF ITS SUB-PHRASES
B12
1
SET SBCNT=0
2
PICK UP EACH SUB-PHRASE
RNT 4 ACCEPT
NO
SBCNT = SBCNT+1
2
YES
MORE SUB-PHRASES
NO
SBCNT? NO.REQUIRED FOR DESCR
YES
"02" 2 IN CODED VALUE OF SUBPHRASES
STORE IN MAIN PHRASE AND POINTERS TO SUB-PHRASES
NO
$TAPE-10P REJECTED
B12
$TAPE-10P ACCEPTED
B8

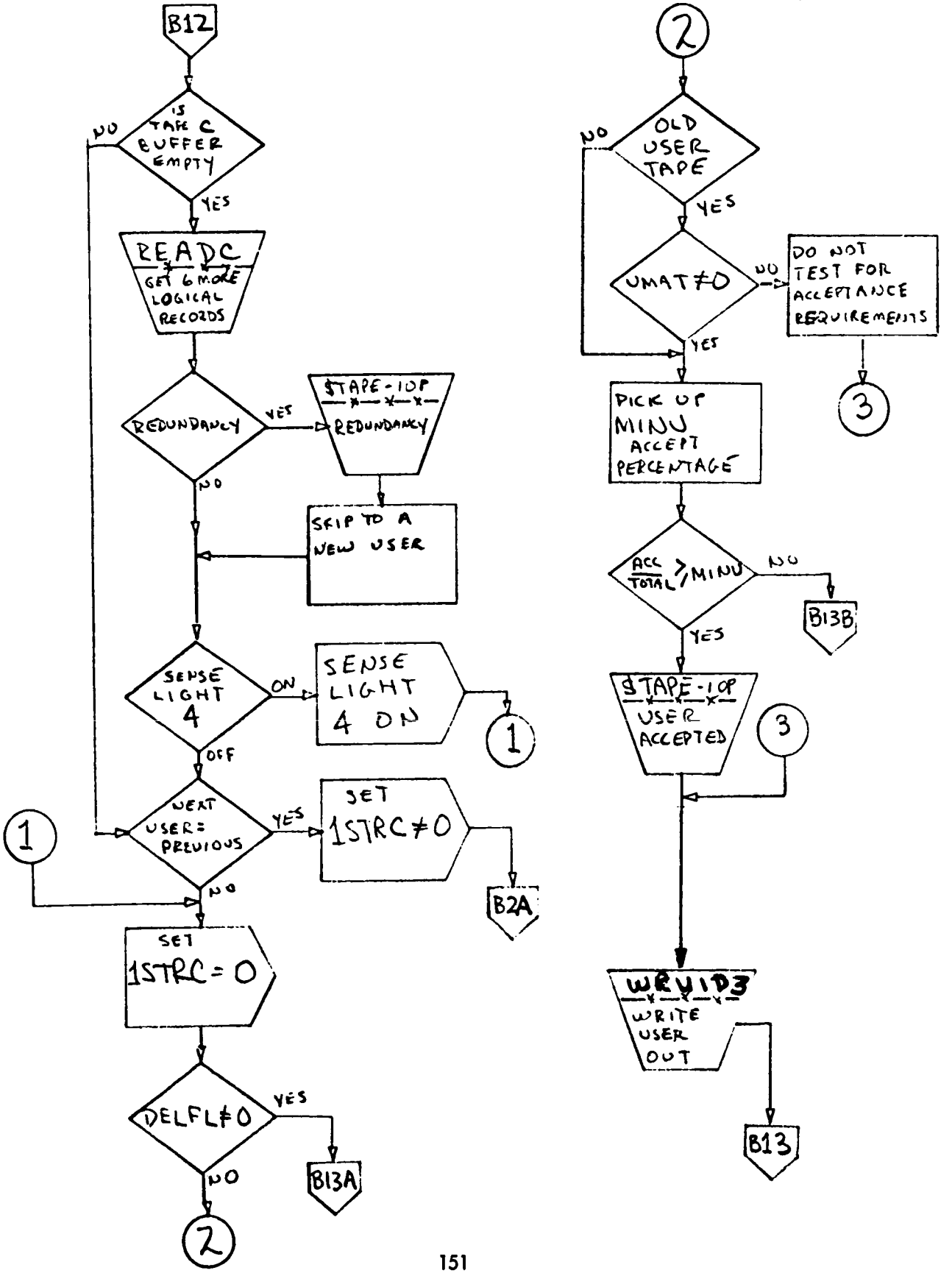

B8
STORE DATE IN HEADER +25 LAST CHANGE DATE
BLDHP=0
NO
YES
UPHP≠0
NO HISTOR.
B12
YES
HMAT=0
NO
$TAPE-IOP
NOT ON HIST. PF TAPE
B12
YES
GET LENGTH OF HIST. PF. LIST OHLEN-26
OHLEN-2670
NO
1
YES
CHECK HIST. PROFILE RECORD
1
FOUND ON H.P. TAPE
NO
YES
B12

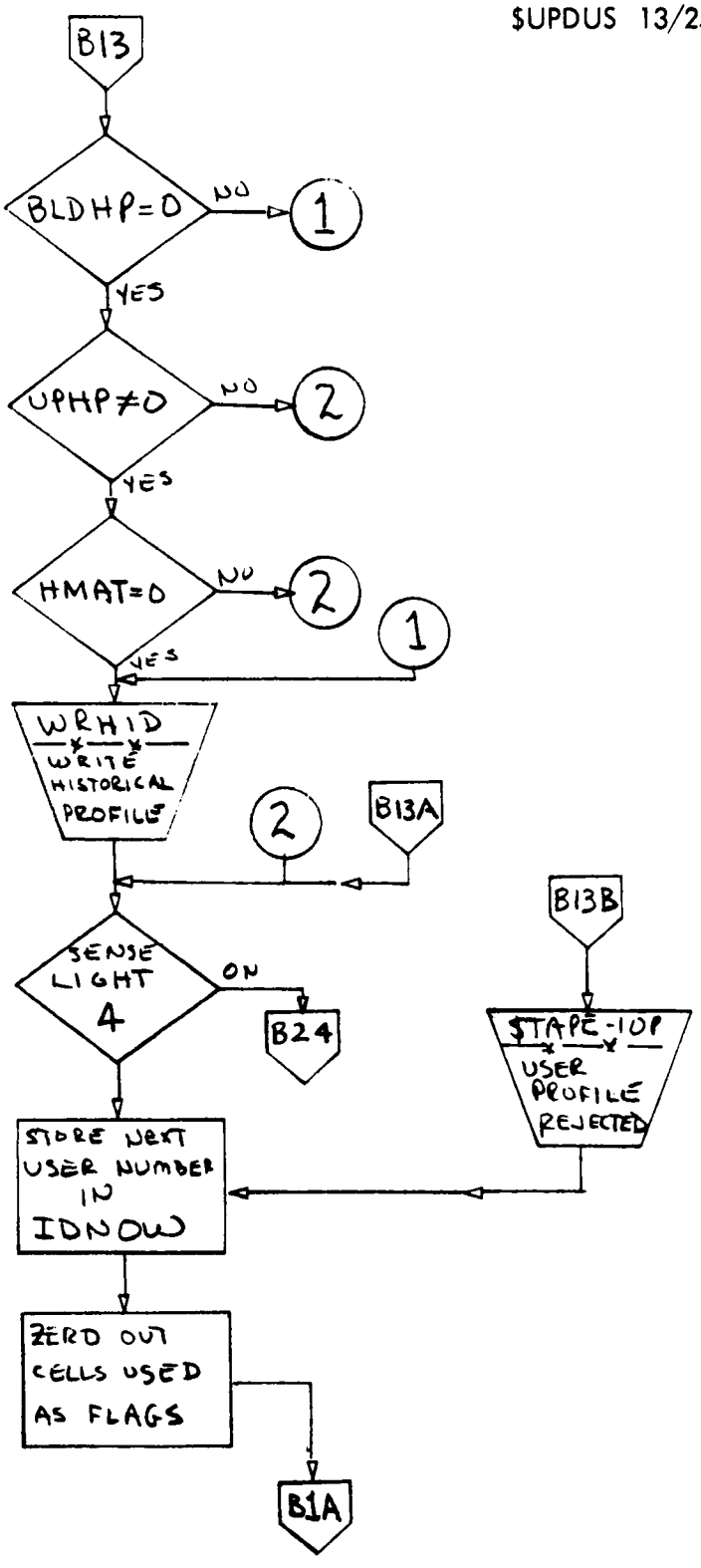

1
STORE CODED VALUE, DATE, IN H.P. RECORD
FIND NON-BLANK LENGTH OF ALPHANUMERIC DESC
STORE NON-BLANK H.P. DESCRIPTOR
STORE LOGICAL RECORD LENGTH
$TAPE-IOP
ADDED TO H.P. TAPE
B12

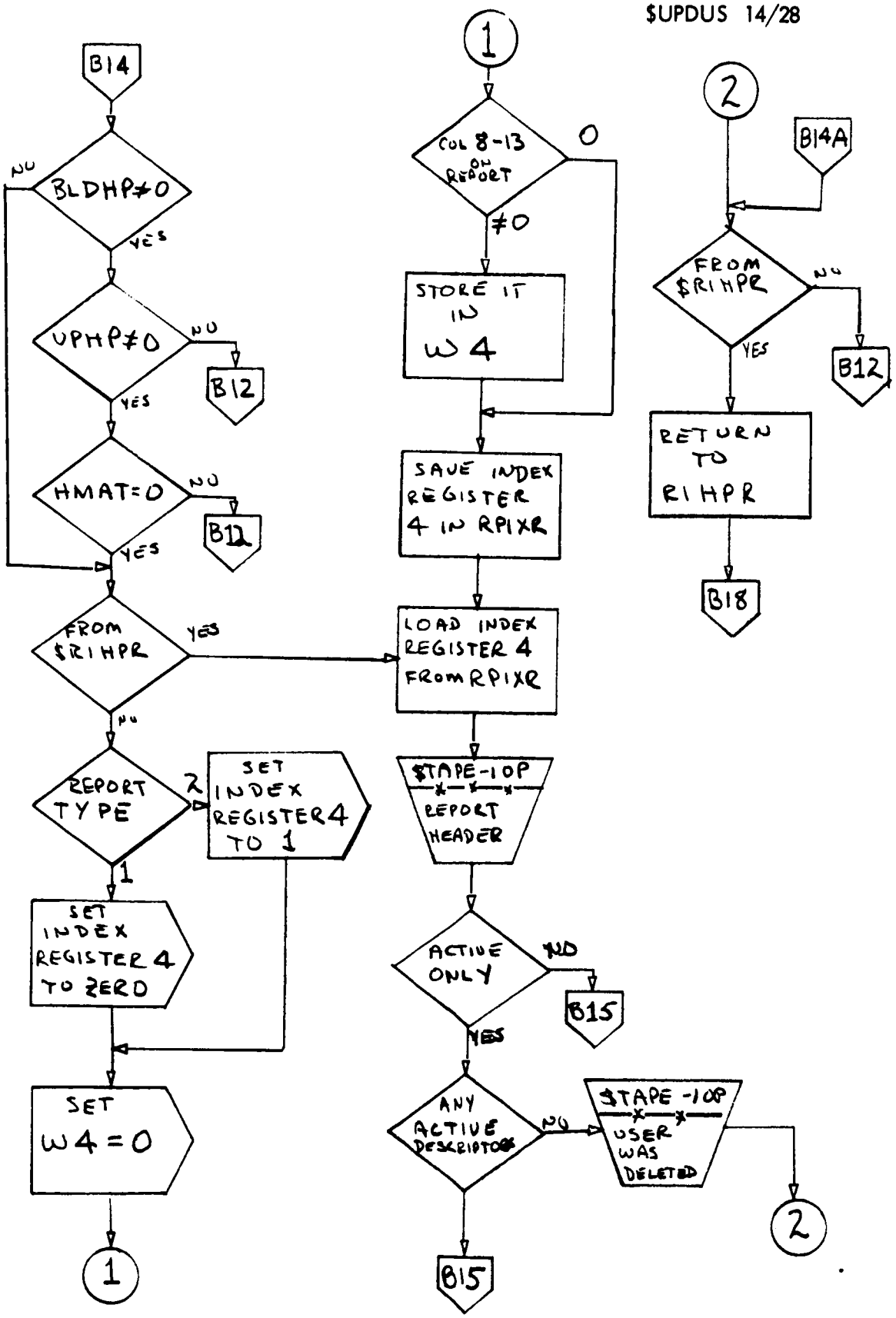

B9
$TAPE-IOP
DESCRIPTOR
. . .
CONVERT CODED VALUE TO BCD FORMAT
OLD UBER TAPE
NO
$TAPE-IOP
DELETE NOT PROCESSED
B12
YES
USER ON OLD USER TAPE
NO
YES
RNT 10
NO
1
YES
SPECIAL
DESCR. ON TAPE
YES
REMOVE 5 OR 6 FROM CODED VALUE IF SINGLE OR MULTI-WORD DESCR
NO
$TAPE-IOP
NOT ON TAPE
B12
1
SINGLE WORD DESCR.
NO
B10
YES
INCREMENT DELETED DESCRIPTOR COUNTER BY 1
REMOVE BIT 35 FROM WORD ON LIST
$TAPE-IOP
DELETED
STORE INDICATORS
LOAD INDICATORS WITH MATCH WORD FROM LIST
RFT 2
B11
ADD 1 TO L OF L,M,N WORD
B11

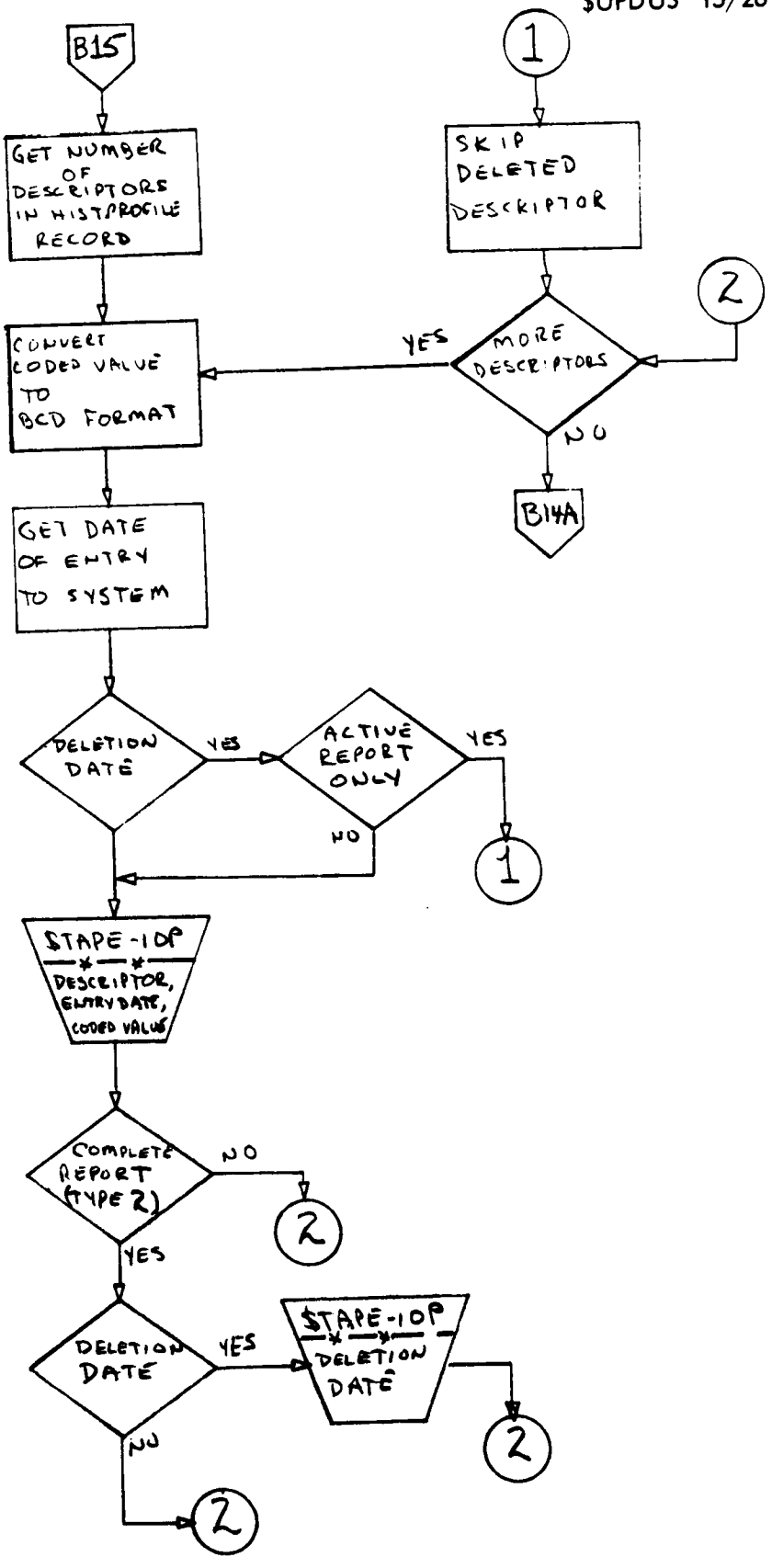

B10
DESCR. FOUND ON PHRASE LIST
NO
$TAPE-IOP
CANNOT DELETE
B11
YES
REMOVE CODED VALUE FROM PHRASE LIST LEAVING 6 ONLY FIRST & LAST 5 BITS
$TAPE-IOP
DELETED
B11

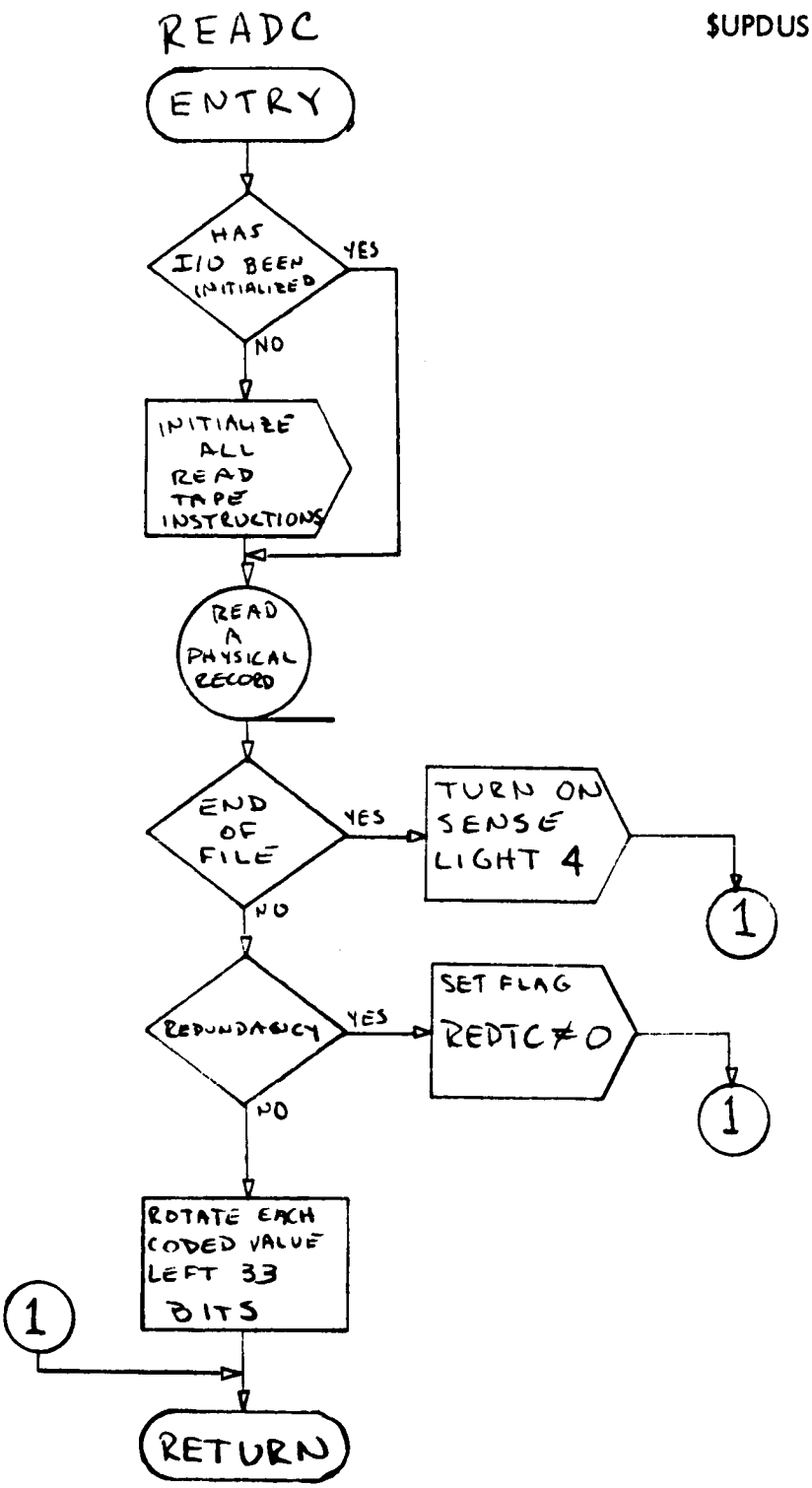

B11
MULTI WORD DESCRIPTOR
NO
YES
SKIP CODED VALUES OF SUB PHRASES ON TAPE C
UPHP≠0
NO
YES
B12
HMAT=0
NO
YES
B12
SEARCH H.P. FOR CODED VALUE TO BE DELETED
FOUND
NO
$TAPE-IOP
NOT ON H.P. TAPE
B12
YES
STORE DELETE DATE
$TAPE-IOP
DELETED FROM H.P.
B12

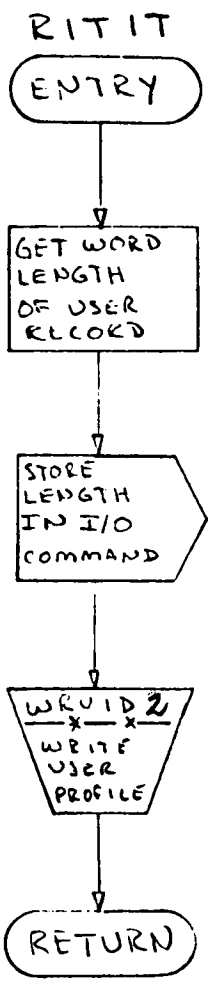

B12
IS TAPE C BUFFER EMPTY
NO
YES
READC
GET 6 MORE LOGICAL RECORDS
REDUNDANCY
YES
$TAPE-IOP
REDUNDANCY
NO
SKIP TO A NEW USER
SENSE LIGHT 4
ON
SENSE LIGHT 4 ON
1
OFF
NEXT USER = PREVIOUS
YES
SET 1STRC≠0
B2A
1
NO
SET 1STRC=0
DELFL≠0
YES
B13A
NO
2
2
OLD USER TAPE
NO
YES
UMAT≠0
NO
DO NOT TEST FOR ACCEPTANCE REQUIREMENTS
3
YES
PICK UP MINU ACCEPT PERCENTAGE
ACC/TOTAL > MINU
NO
B13B
YES
$TAPE-IOP
USER ACCEPTED
3
WRUID3
WRITE USER OUT
B13

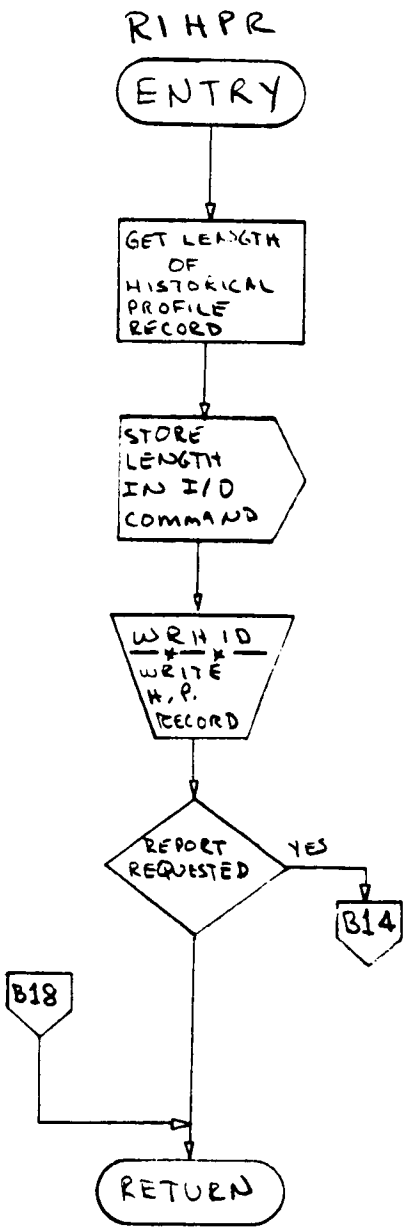

B13
BLDHP=0
NO
1
YES
UPHP≠0
NO
2
YES
HMAT=0
NO
2
1
YES
WRHID
WRITE HISTORICAL PROFILE
2
B13A
B13B
SENSE LIGHT 4
ON
B24
$TAPE-IOP
USER PROFILE REJECTED
STORE NEXT USER NUMBER IN IDNOW
ZERO OUT CELLS USED AS FLAGS
B1A

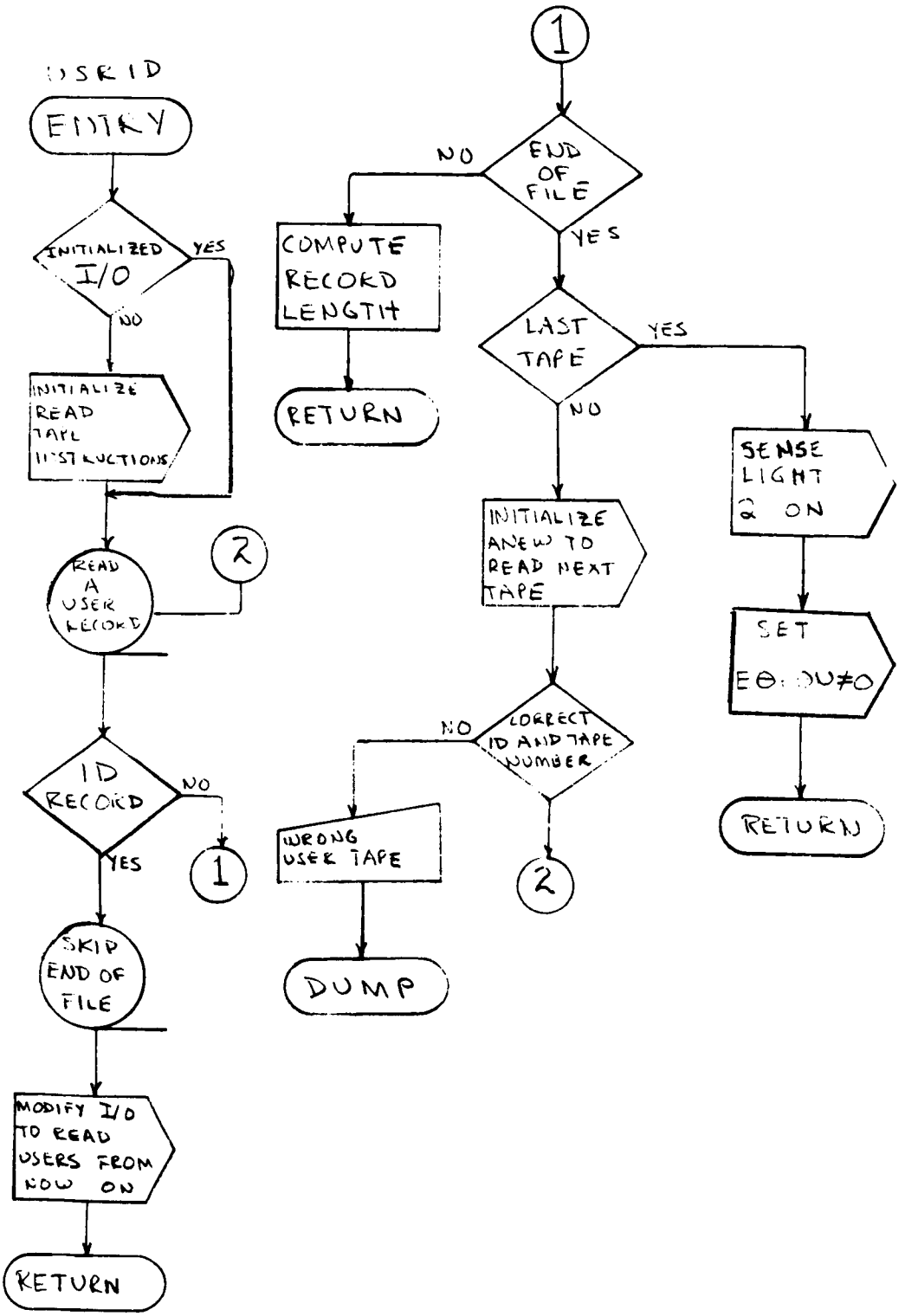

B14
BLDHP≠0
NO
YES
UPHP≠0
NO
B12
YES
HMAT=0
NO
B11
YES
FROM $RIHPR
YES
NO
REPORT TYPE
2
SET INDEX REGISTER 4 TO 1
1
SET INDEX REGISTER 4 TO ZERO
SET W4=0
1
1
COL 8-13 ON REPORT
0
≠0
STORE IT IN W4
SAVE INDEX REGISTER 4 IN RPIXR
LOAD INDEX REGISTER 4 FROM RPIXR
$TAPE-IOP
REPORT HEADER
ACTIVE ONLY
NO
B15
YES
ANY ACTIVE DESCRIPTORS
NO
$TAPE-IOP
USER WAS DELETED
2
B15
2
B14A
FROM $RIHPR
NO
B12
YES
RETURN TO RIHPR
B18

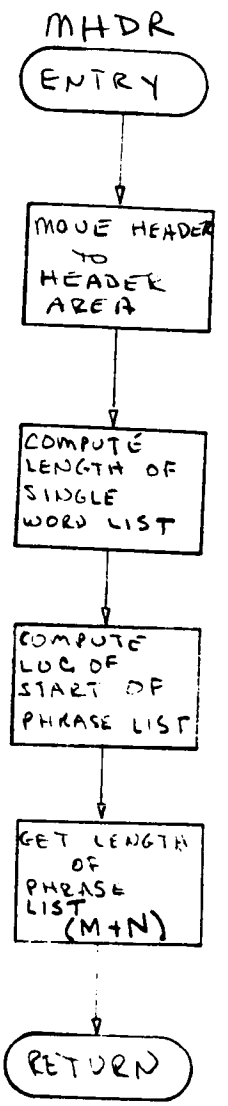

B15
GET NUMBER OF DESCRIPTORS IN HIST/PROFILE RECORD
CONVERT CODED VALUE TO BCD FORMAT
GET DATE OF ENTRY TO SYSTEM
DELETION DATE
YES
ACTIVE REPORT ONLY
YES
NO
1
$TAPE-IOP
DESCRIPTOR, ENTRY DATE, CODED VALUE
COMPLETE REPORT (TYPE 2)
NO
2
YES
DELETION DATE
YES
$TAPE-IOP
DELETION DATE
2
NO
2
1
SKIP DELETED DESCRIPTOR
2
MORE DESCRIPTORS
YES
NO
B14A

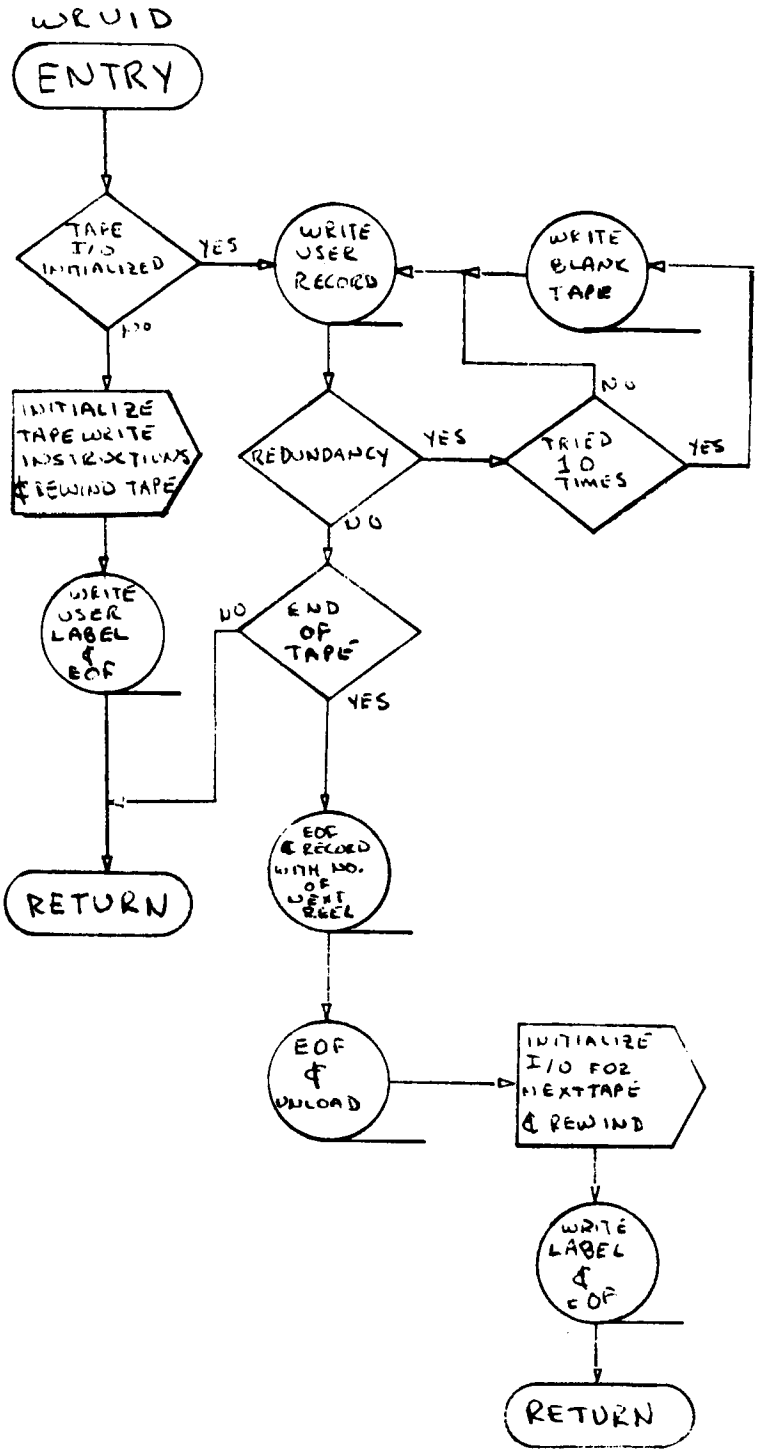

READC
ENTRY
HAS I/O BEEN INITIALIZED
YES
NO
INITIALIZE ALL READ TAPE INSTRUCTIONS
READ A PHYSICAL RECORD
END OF FILE
YES
TURN ON SENSE LIGHT 4
1
NO
REDUNDANCY
YES
SET FLAG REDTC≠0
1
NO
ROTATE EACH CODED VALUE LEFT 33 BITS
1
RETURN

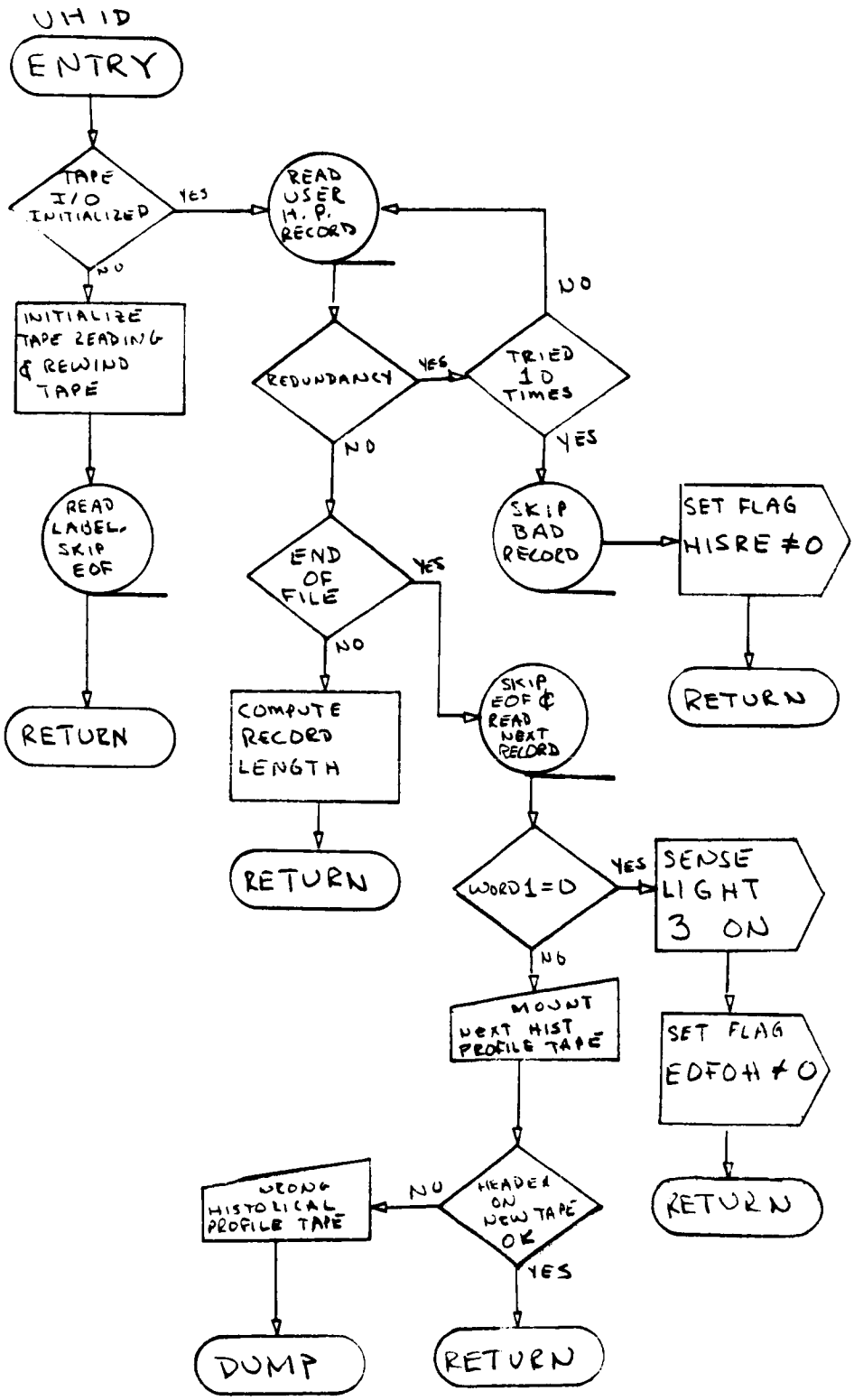

RITIT
ENTRY
GET WORD LENGTH OF USER RLCOKD
STORE LENGTH IN I/O COMMAND
WRUID 2
WRITE USER PROFILE
RETURN

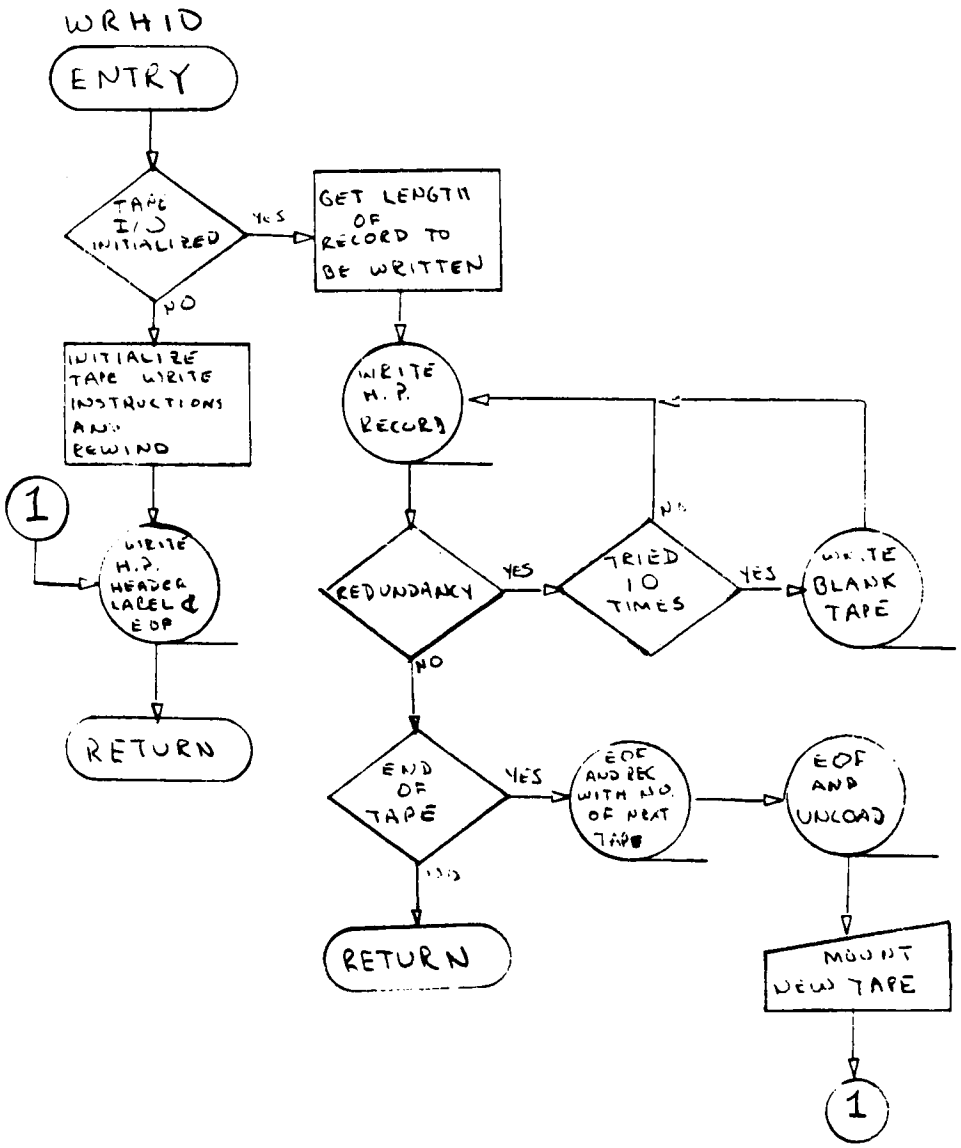

RIHPR
ENTRY
GET LENGTH OF HISTORICAL PROFILE RECORD
STORE LENGTH IN I/O COMMAND
WRITE H.P. RECORD
REPORT REQUESTED
YES
B14
B18
RETURN

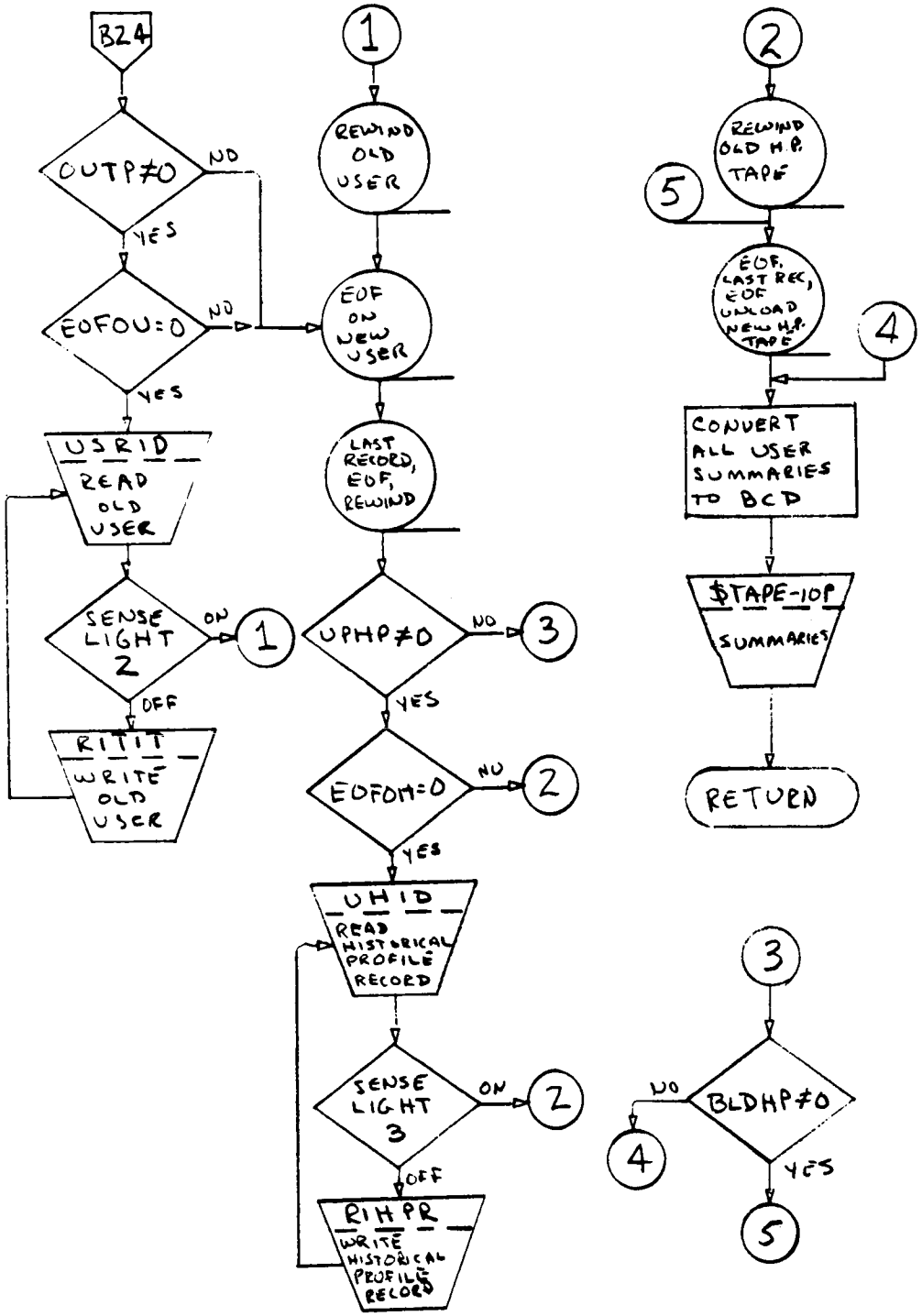

USRID
ENTRY
INITIALIZED I/O
YES
NO
INITIALIZE READ TAPE INSTRUCTIONS
READ A USER RECORD
2
ID RECORD
NO
YES
1
SKIP END OF FILE
MODIFY I/O TO READ USERS FROM NOW ON
RETURN
1
END OF FILE
NO
YES
COMPUTE RECORD LENGTH
RETURN
LAST TAPE
YES
NO
INITIALIZE ANEW TO READ NEXT TAPE
CORRECT ID AND TAPE NUMBER
NO
WRONG USER TAPE
DUMP
2
SENSE LIGHT 2 ON
SET EOF DU≠0
RETURN

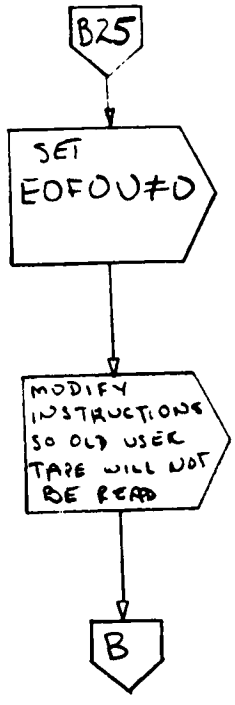

MHDR
ENTRY
MOVE HEADER TO HEADER AREA
COMPUTE LENGTH OF SINGLE WORD LIST
COMPUTE LOC OF START OF PHRASE LIST
GET LENGTH OF PHRASE LIST (M+N)
RETURN

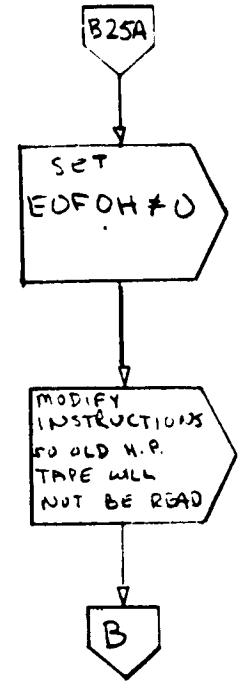

WRUID
ENTRY
TAPE I/O INITIALIZED
YES
NO
WRITE USER RECORD
WRITE BLANK TAPE
INITIALIZE TAPE WRITE INSTRUCTIONS & REWIND TAPE
REDUNDANCY
YES
NO
TRIED 10 TIMES
YES
NO
WRITE USER LABEL & EOF
NO
END OF TAPE
YES
RETURN
EOF & RECORD WITH NO. OF NEXT REEL
EOF & UNLOAD
INITIALIZE I/O FOR NEXT TAPE & REWIND
WRITE LABEL & EOF
RETURN

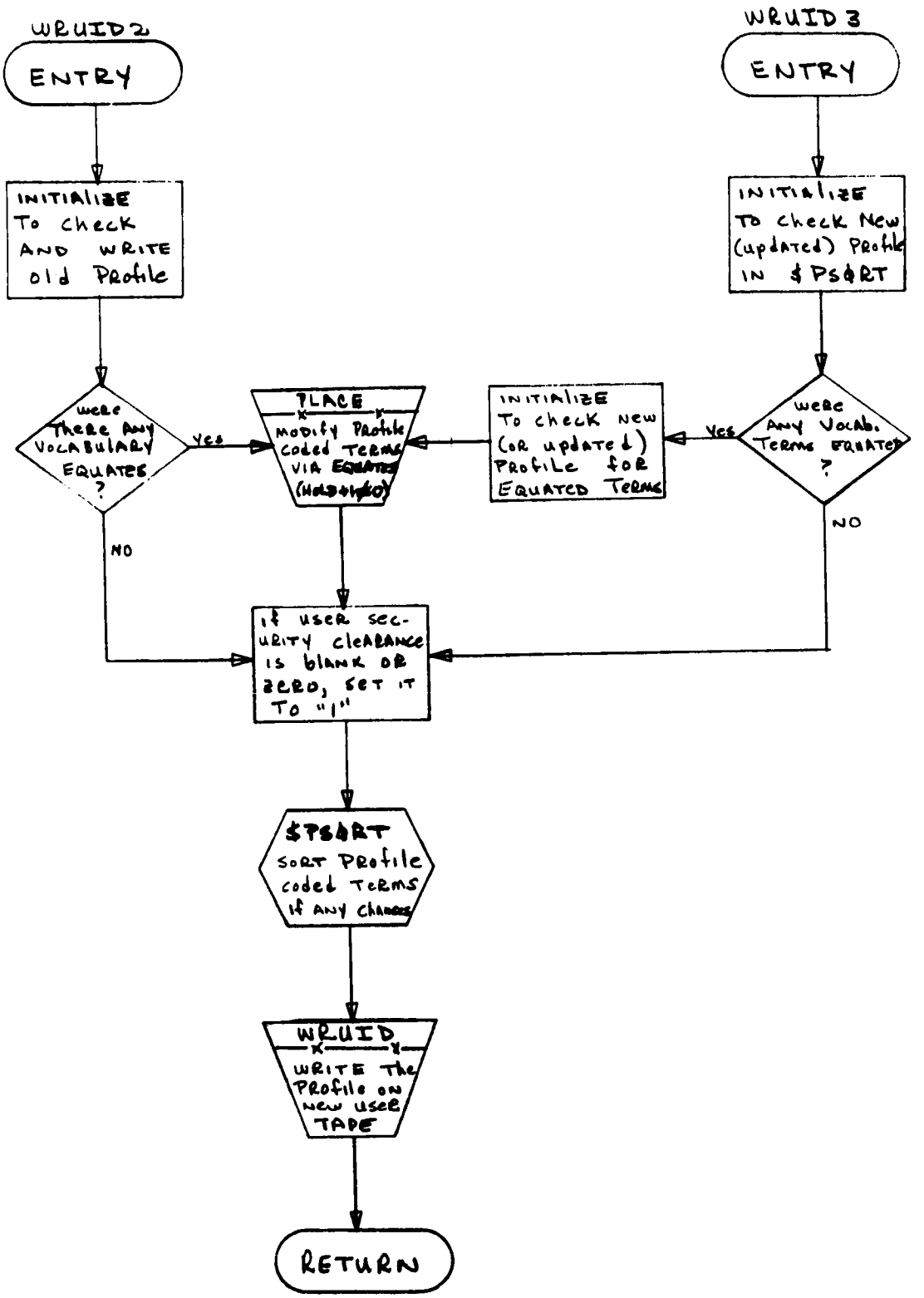

UH ID
ENTRY
TAPE I/O INITIALIZED
YES
NO
READ USER H. P. RECORD
INITIALIZE TAPE READING & REWIND TAPE
REDUNDANCY
YES
NO
TRIED 10 TIMES
NO
YES
READ LABEL, SKIP EOF
END OF FILE
YES
NO
SKIP BAD RECORD
SET FLAG HISRE ≠ 0
RETURN
RETURN
COMPUTE RECORD LENGTH
SKIP EOF & READ NEXT RECORD
RETURN
WORD 1 = 0
YES
NO
SENSE LIGHT 3 ON
MOUNT NEXT HIST PROFILE TAPE
SET FLAG EOFOH ≠ 0
RETURN
WRONG HISTOLICAL PROFILE TAPE
NO
HEADER ON NEW TAPE OK
YES
DUMP
RETURN

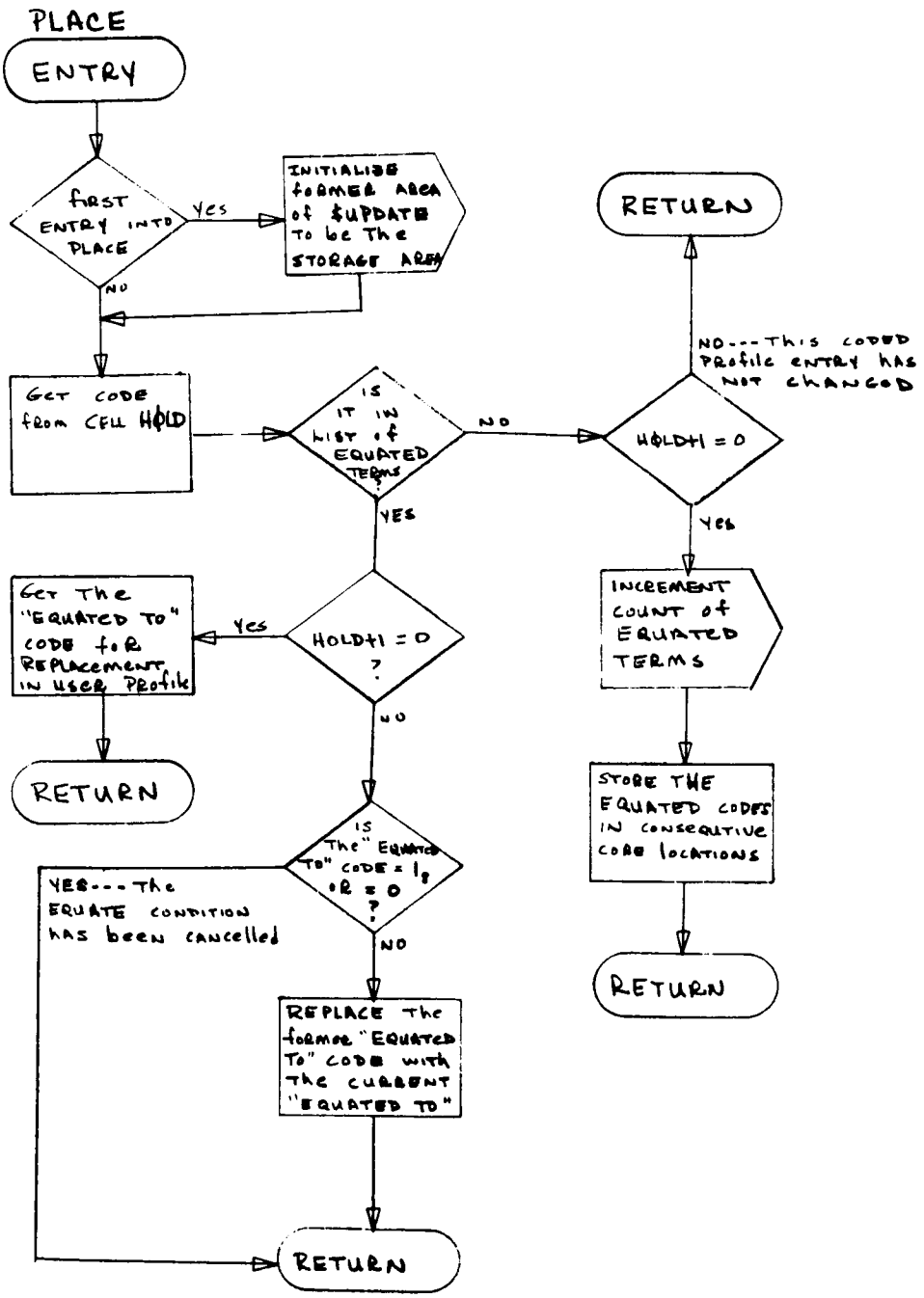

WRHID
ENTRY
TAPE I/O INITIALIZED
YES
NO
GET LENGTH OF RECORD TO BE WRITTEN
INITIALIZE TAPE WRITE INSTRUCTIONS AND REWIND
WRITE H.P. RECORD
1
WRITE H.P. HEADER LABEL & EOF
REDUNDANCY
YES
NO
TRIED 10 TIMES
YES
NO
WRITE BLANK TAPE
RETURN
END OF TAPE
YES
NO
EOF AND REC WITH NO. OF NEXT TAPE
EOF AND UNLOAD
MOUNT NEW TAPE
RETURN
1

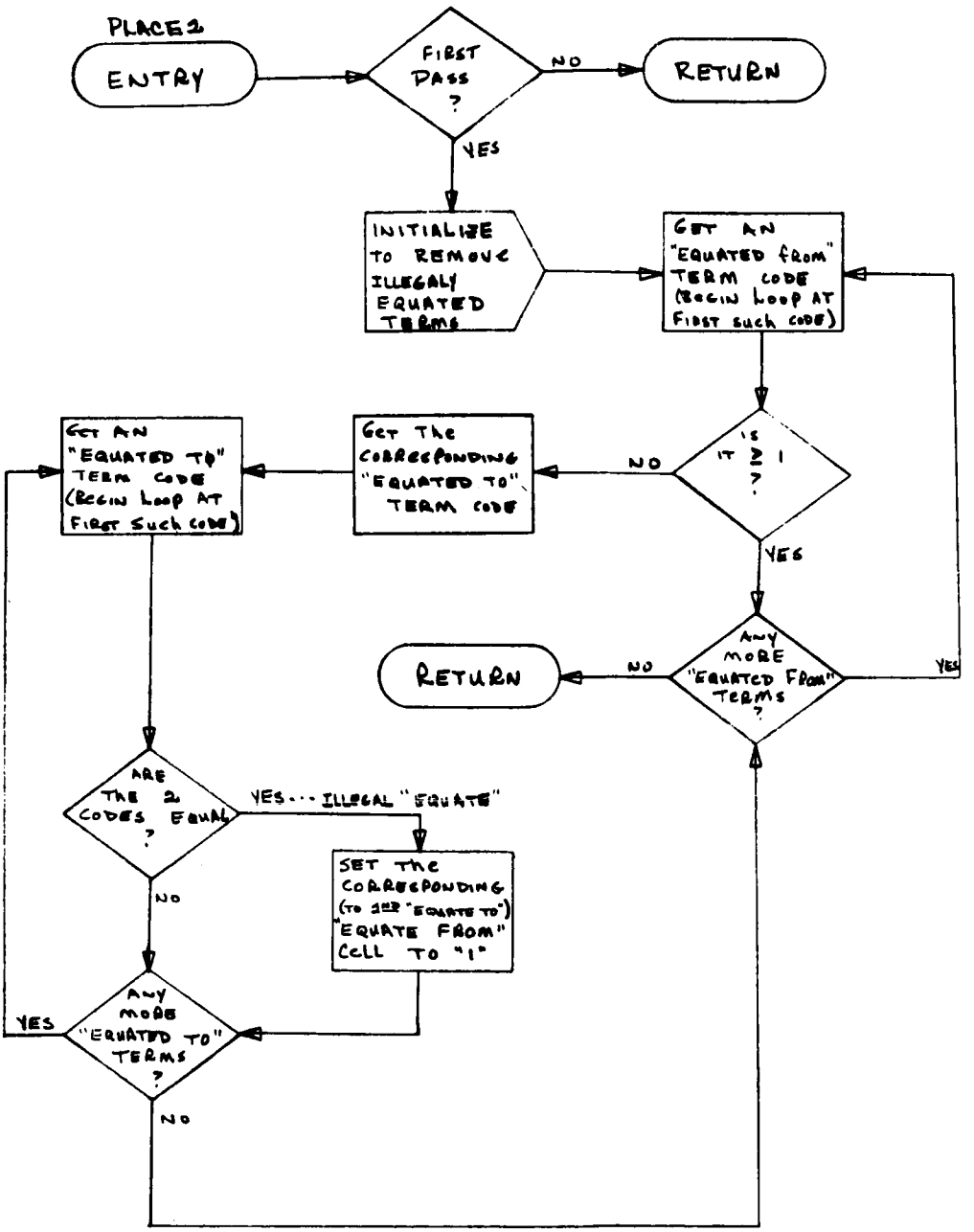

B24
OUTP≠0
NO
YES
EOFOU=0
NO
YES
USRID
READ OLD USER
SENSE LIGHT 2
ON
1
OFF
RITIT
WRITE OLD USER
1
REWIND OLD USER
EOF ON NEW USER
LAST RECORD, EOF, REWIND
UPHP≠0
NO
3
YES
EOFOM=0
NO
2
YES
UHID
READ HISTORICAL PROFILE RECORD
SENSE LIGHT 3
ON
2
OFF
RIHPR
WRITE HISTORICAL PROFILE RECORD
2
REWIND OLD M.P. TAPE
5
EOF, LAST REC, EOF UNLOAD NEW M.P. TAPE
4
CONVERT ALL USER SUMMARIES TO BCD
$TAPE-IOP
SUMMARIES
RETURN
3
BLDHP≠0
NO
4
YES
5


B25
SET
EOFOU≠0
MODIFY
INSTRUCTIONS
SO OLD USER
TAPE WILL NOT
BE READ
B


B25A
SET
EOFOH≠0
MODIFY
INSTRUCTIONS
SO OLD H.P.
TAPE WILL
NOT BE READ
B

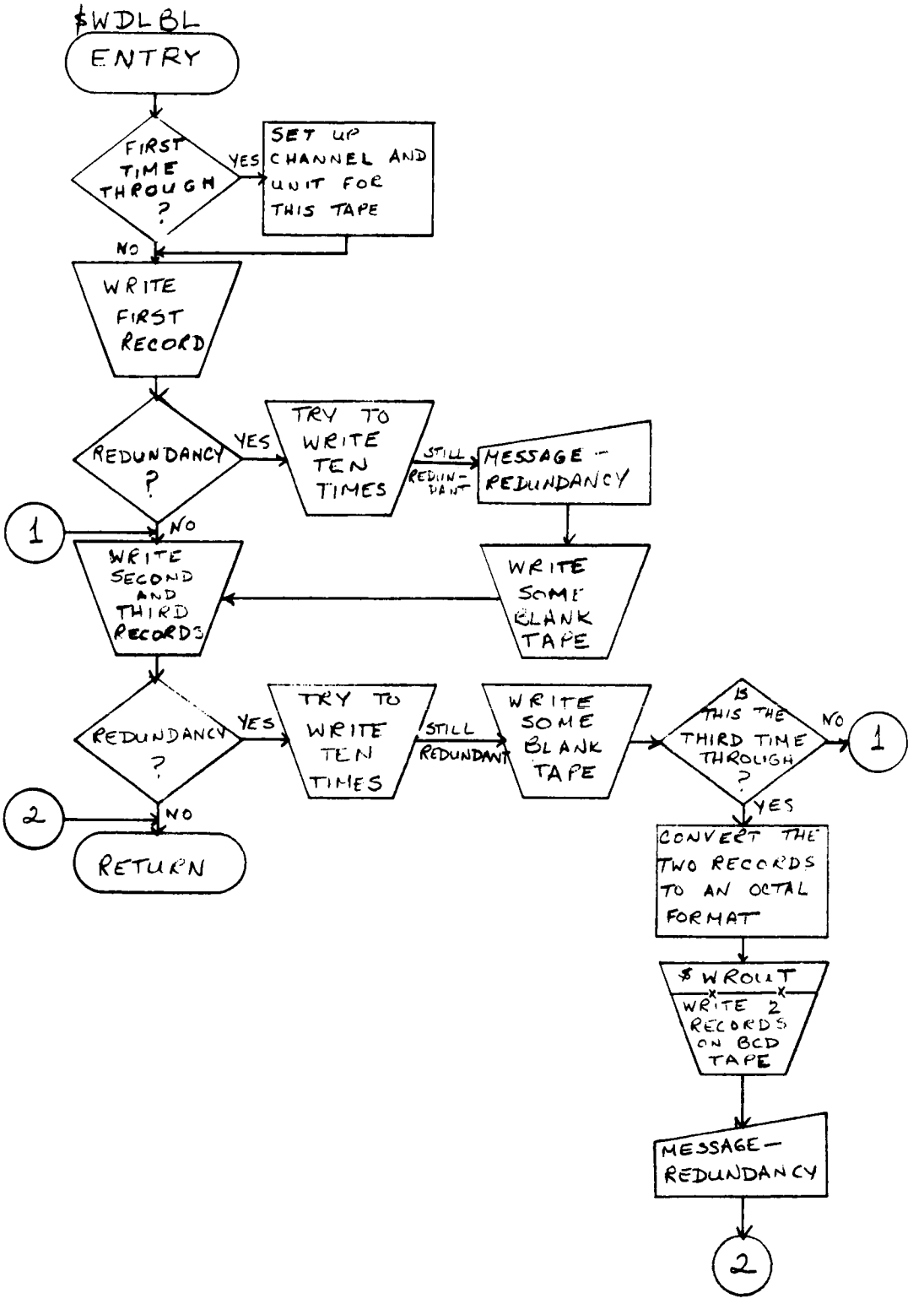

WRUID 2
ENTRY
INITIALIZE To check AND WRITE old Profile
WRUID 3
ENTRY
INITIALIZE To check New (updated) Profile IN PSRT
WERE THERE ANY VOCABULARY EQUATES ?
Yes
NO
PLACE
MODIFY Profile coded TERMS VIA EQUATES
INITIALIZE To check NEW (OR updated) Profile FOR EQUATED TERMS
Yes
WERE ANY Vocab. Terms EQUATED ?
NO
IF USER SECURITY CLEARANCE IS BLANK OR ZERO, SET IT TO "1"
TSRT
SORT Profile coded TERMS IF ANY Changes
WRUID
WRITE The PROFILE ON NEW USER TAPE
RETURN


PLACE
ENTRY
first ENTRY INTO PLACE
Yes
INITIALISE former AREA of $UPDATE To be The STORAGE AREA
NO
Get CODE from CELL HOLD
IS IT IN LIST of EQUATED TERMS ?
NO
HOLD+1 = 0
RETURN
NO---This CODED Profile ENTRY HAS NOT CHANGED
YES
YES
INCREMENT COUNT of EQUATED TERMS
STORE THE EQUATED CODES IN CONSECUTIVE CORE LOCATIONS
RETURN
HOLD+1 = 0 ?
Yes
Get The "EQUATED TO" CODE for REPLACEMENT IN USER Profile
RETURN
NO
IS "The" EQUATED TO" CODE = 1 OR = 0 ?
YES---The EQUATE CONDITION has been CANCELLED
NO
REPLACE The former "EQUATED TO" CODE with The CURRENT "EQUATED TO"
RETURN


PLACE 2
ENTRY
FIRST PASS ?
NO
RETURN
YES
INITIALIZE TO REMOVE ILLEGALLY EQUATED TERMS
GET AN "EQUATED FROM" TERM CODE (BEGIN LOOP AT FIRST SUCH CODE)
IS IT ≥ 1 ?
NO
GET THE CORRESPONDING "EQUATED TO" TERM CODE
GET AN "EQUATED TO" TERM CODE (BEGIN LOOP AT FIRST SUCH CODE)
YES
ANY MORE "EQUATED FROM" TERMS ?
NO
RETURN
YES
ARE THE 2 CODES EQUAL ?
YES... ILLEGAL "EQUATE"
SET THE CORRESPONDING (TO 1ST "EQUATE TO") "EQUATE FROM" CELL TO "1"
NO
ANY MORE "EQUATED TO" TERMS ?
YES
NO

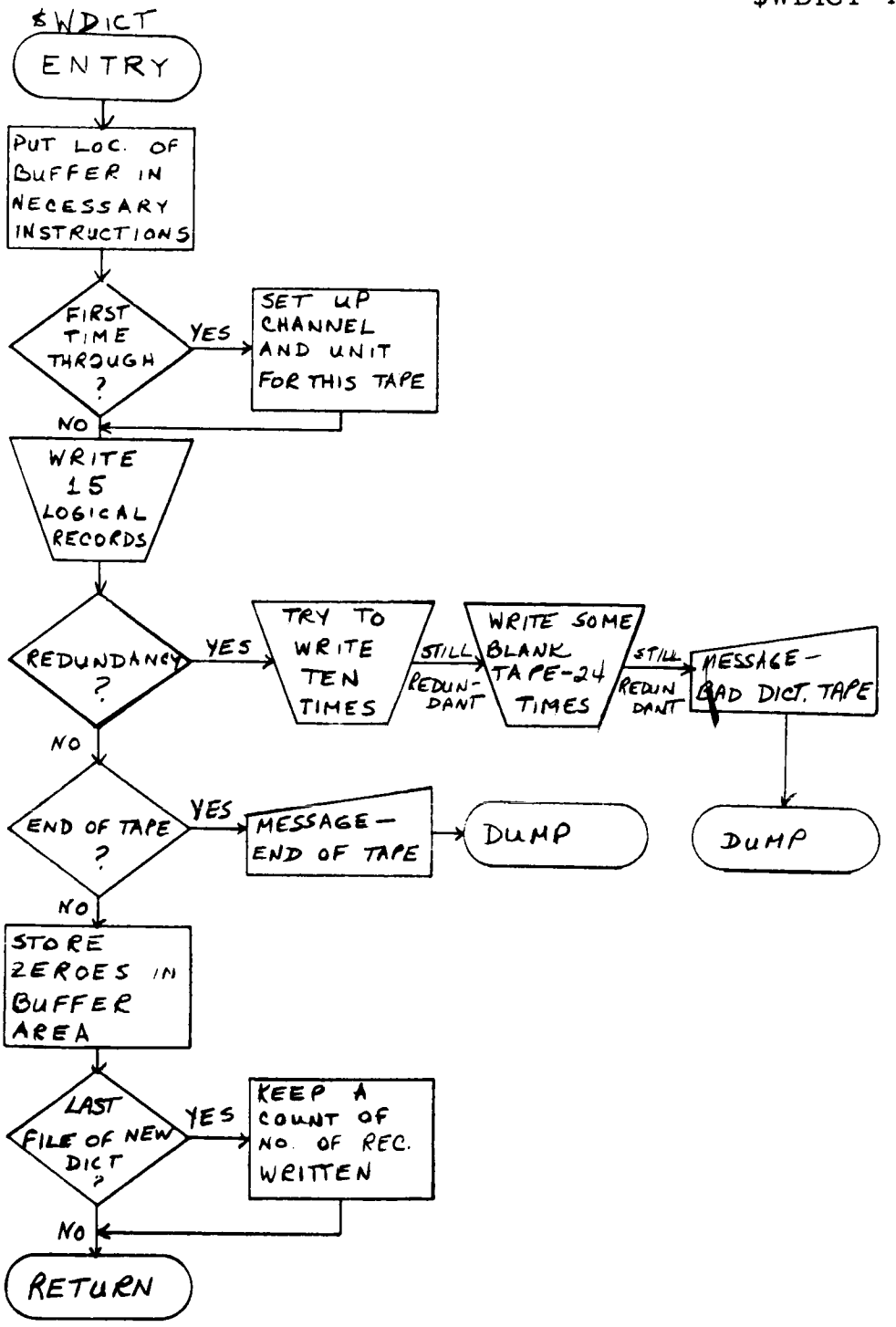
Subroutine WDLBL

This subroutine writes the first file of the new vocabulary. The first record contains the tape label. If there is a redundancy, an attempt is made to write the record ten times. If unsuccessful, a message indicating the redundancy condition is printed and tape is spaced forward. The second record, which contains the catalog of minimum and maximum entries, and the third record, which contains a table of the number of logical records per file, are written next. If there is a redundancy, an attempt is made to write the records ten times. If the redundancy persists, some blank tape is written, and an attempt is made to write the two records again. If the redundancy still exists, one more try is made writing blank tape and writing the records. If there is still a redundancy, the records are written on the system output tape with twelve octal numbers for each computer word.


$WDLBL
ENTRY
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE FIRST RECORD
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
MESSAGE - REDUNDANCY
WRITE SOME BLANK TAPE
1
NO
WRITE SECOND AND THIRD RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
WRITE SOME BLANK TAPE
IS THIS THE THIRD TIME THROUGH ?
NO
1
YES
CONVERT THE TWO RECORDS TO AN OCTAL FORMAT
$WROUT
WRITE 2 RECORDS ON BCD TAPE
MESSAGE - REDUNDANCY
2
2
NO
RETURN

Subroutine WDICT

This subroutine writes the new vocabulary descriptor records, which are blocked fifteen logical records per physical record. The procedures for a redundancy condition and an end-of-tape condition are the same as those used in WDOCT. A count of the number of records in the last file of the new vocabulary is kept for use by the subroutine UPDATE.


$WDICT
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 15 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
WRITE SOME BLANK TAPE-24 TIMES
STILL REDUN DANT
MESSAGE — BAD DICT. TAPE
DUMP
NO
END OF TAPE ?
YES
MESSAGE — END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
LAST FILE OF NEW DICT ?
YES
KEEP A COUNT OF NO. OF REC. WRITTEN
NO
RETURN

Subroutine WDOC

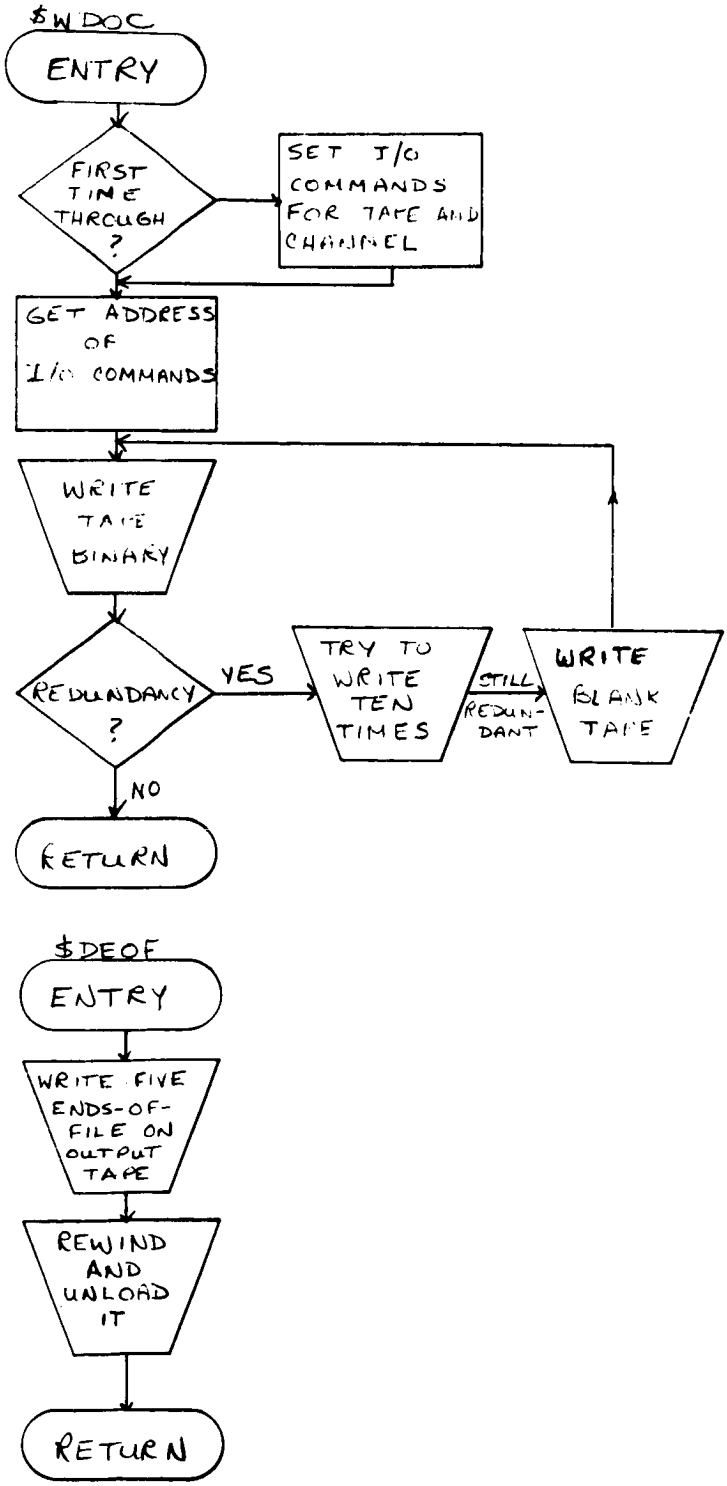
WDOC writes the binary document profiles tape. It is entered from subroutine PDTB. If a redundancy occurs while writing, ten attempts will be made, after which blank tape will be written. Ten more attempts will be made.

Subroutine DEOF

DEOF writes five EOF's on the binary document profiles tape. It then rewinds and unloads the tape.

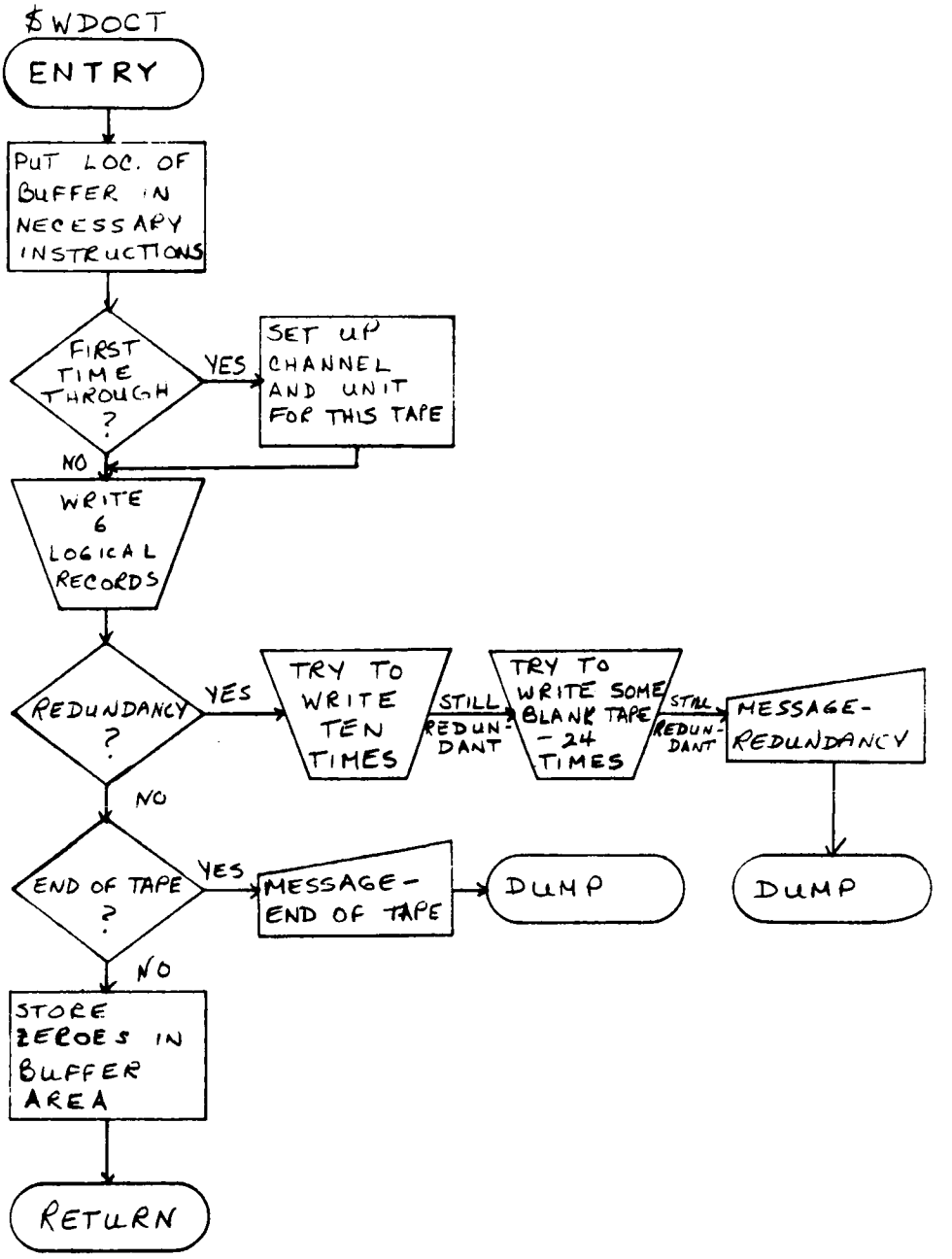
$WDOC 1/1

$DEOF 1/1



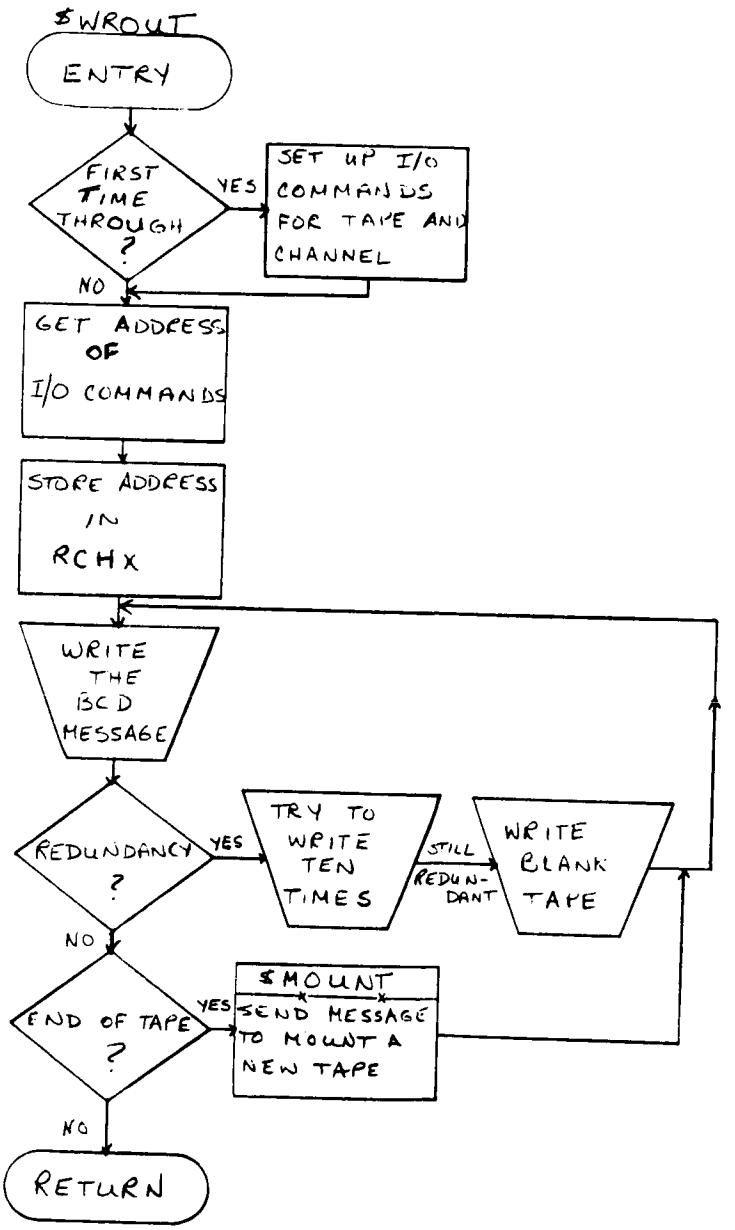
Subroutine WDOCT

This subroutine writes the coded document tape, which is blocked six logical records per physical record. For a redundant record, an attempt to write is made ten times. If the redundancy persists, blank tape is written and another attempt to write is made. This process can occur 24 times. If, finally, there is still a redundancy, a message is printed and a dump of core is taken. A message and a dump also will be given for an EOT.


$WDOCT
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
TRY TO WRITE SOME BLANK TAPE - 24 TIMES
STILL REDUNDANT
MESSAGE-REDUNDANCY
DUMP
NO
END OF TAPE ?
YES
MESSAGE-END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
RETURN

Subroutine WROUT

All messages written on the system output tape in PDTB are handled by this subroutine. The output instructions are initialized the first time through. The exception is the RCH instruction, which must be set up before each TSX to WROUT.

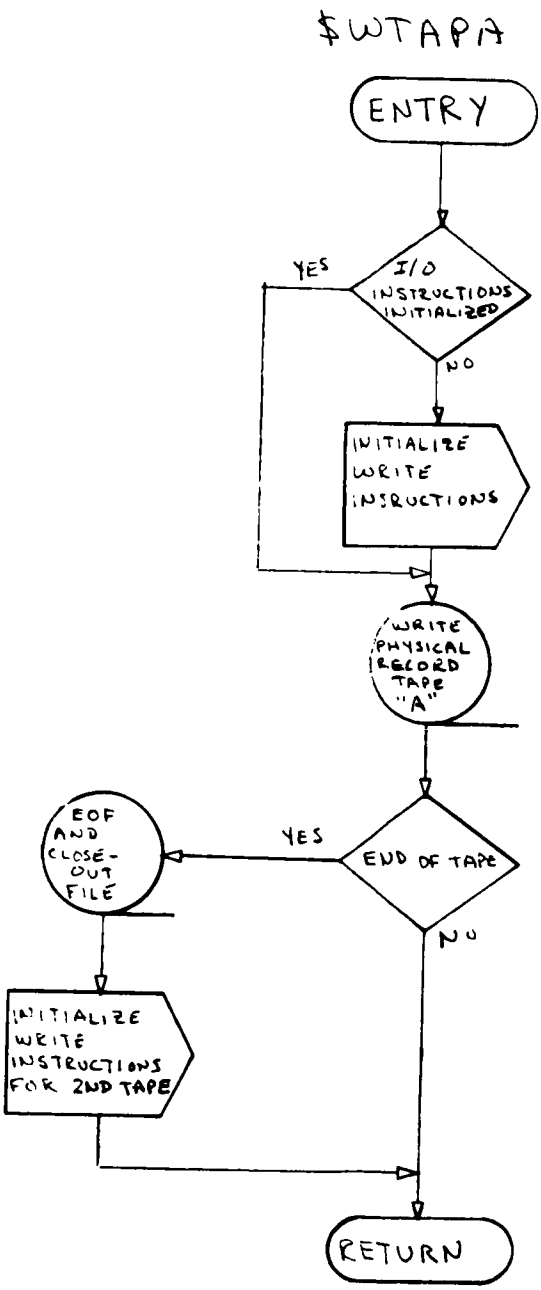

$WROUT
ENTRY
FIRST TIME THROUGH ?
YES
SET UP I/O COMMANDS FOR TAPE AND CHANNEL
NO
GET ADDRESS OF I/O COMMANDS
STORE ADDRESS IN RCHx
WRITE THE BCD MESSAGE
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
WRITE BLANK TAPE
NO
END OF TAPE ?
YES
$MOUNT
SEND MESSAGE TO MOUNT A NEW TAPE
NO
RETURN

Subroutine WTAPA

Whenever six logical records from $CODER have been stored in an area called BLOCK (6 x 27 = 162 words), WTAPA is entered to write these six on tape A as one physical record.

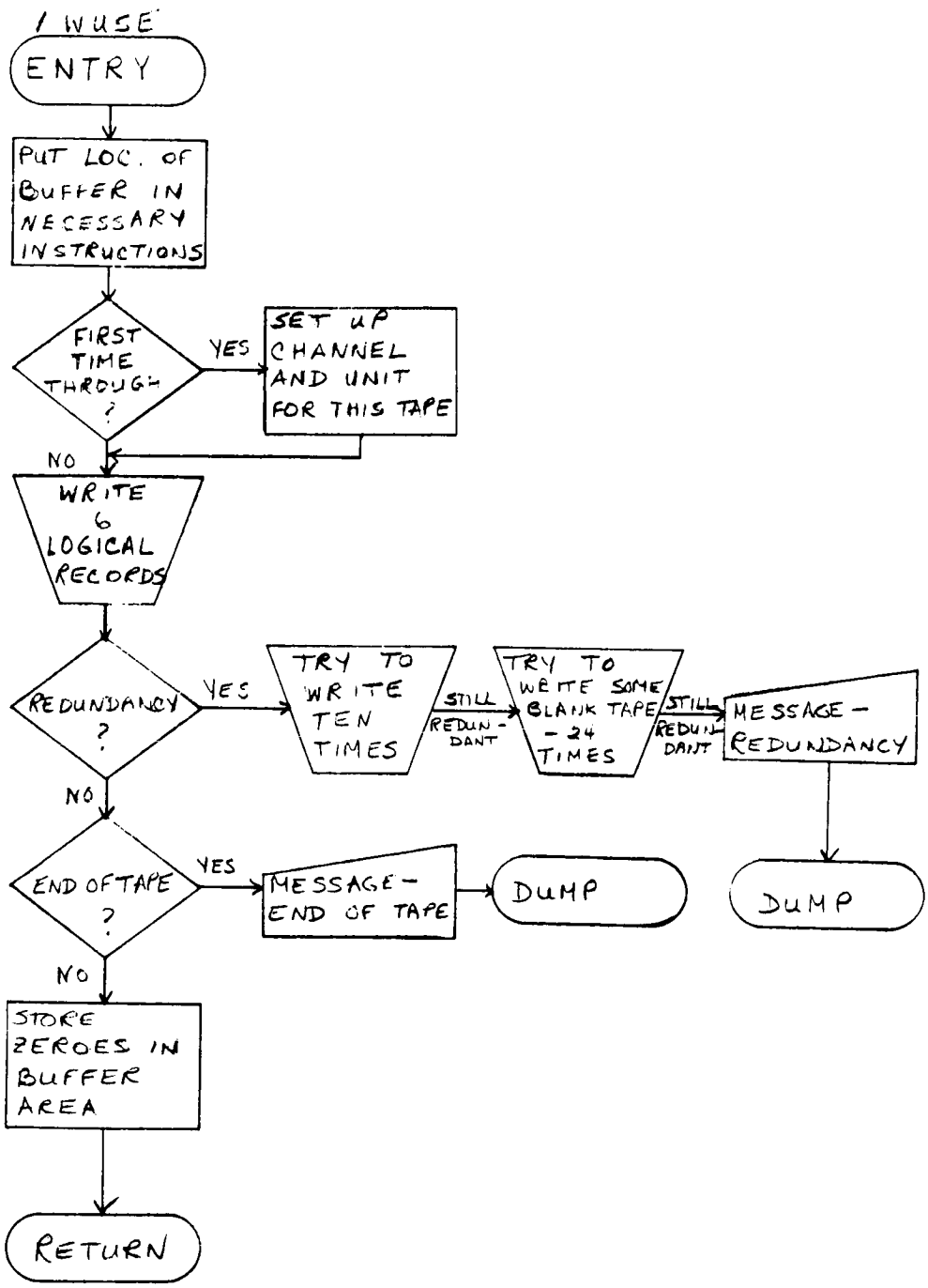
Upon first entry, all output instructions are set up. After the first entry, WTAPA only writes the 162-word binary record. If one tape is filled, an EOF will be written, followed by a 20-word BCD control record for the SORT subroutine, and another EOF. The tape will be rewound, and the second tape A will immediately be put into use.

After every write, blanks will be stored in every word in the BLOCK area, with the exception of the first three locations of each logical record (BLOCK, BLOCK + 1, BLOCK + 2, BLOCK + 27, BLOCK + 28, etc.); they will be filled with zeroes.


$WTAPA
ENTRY
I/O INSTRUCTIONS INITIALIZED
YES
NO
INITIALIZE WRITE INSTRUCTIONS
WRITE PHYSICAL RECORD TAPE "A"
END OF TAPE
YES
NO
EOF AND CLOSE-OUT FILE
INITIALIZE WRITE INSTRUCTIONS FOR 2ND TAPE
RETURN

Subroutine WUSE

This subroutine writes the user tape, which is blocked six logical records per physical record. All processing in this routine is identical to that in the WDOCT subroutine.


/ WUSE
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
TRY TO WRITE SOME BLANK TAPE - 24 TIMES
STILL REDUN-DANT
MESSAGE - REDUNDANCY
DUMP
NO
END OF TAPE ?
YES
MESSAGE - END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
RETURN

C. SDI MATCH PROGRAM

1. Program Summary

1.1 Description

The SDI Profile Matching Program (MATCH) for the IBM 7090/94 Data Processing System matches coded document profiles against coded user profiles, generating notices when match criteria are met. The program runs under the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with three to five 729 tape units (in addition to FORTRAN units). MATCH consists of a primary program written in FORTRAN and of several subroutines, written in FAP.

In SDI-5, profile descriptor matching is a function of matching single-word descriptors, matching or partially matching phrase descriptors, and, in user profiles, 'must' and 'not' modifiers. Specifically, the MATCH program generates a notice of a document for a user when at least one of the following matching criteria is met:

1. A minimum percentage of matching 'may' (unmodified) single-word descriptors.
2. One matching 'must' descriptor.
3. One matching or partially matching phrase descriptor: two words of a two-word phrase, three of three, three of four, three of five, four of six and five of seven.

But, in any of the above cases, if one matching 'not' descriptor, phrase or single-word is also found, no notice is generated; the 'not' phrase descriptor is subject to the conditions in Item 3 and the 'not' single-word descriptor is equivalent but opposite to a 'must'. Also, in each execution of MATCH, a small percentage of notices may be generated at random to give the user an idea of what he may be missing due to a faulty profile.

The primary MATCH program contains the basic logic of the SDI profile matching phase. It reads and acts on the control card of the program parameters, orders profile input activity, determines whether or not a notice is to be sent and generates notices as required. When the primary program reaches statement 200, the FORTRAN monitor is no longer available; it has been replaced by user profiles in the array USER. This is done to allow as large a buffer as possible for user profiles. The size of array USER is determined by subroutine ERASE, which must be "called" at the end, physically, of the main program.

The important primary program variables that enter the program via control

card are:

1.	NTAPD	FORTRAN logical tape unit, document input
2.	NTAPU	FORTRAN logical tape unit, user input reel 1
3.	NTAPU2	FORTRAN logical tape unit, user input reel 2, if any
4.	NOTIC	FORTRAN logical tape unit, notice output reel 1
5.	NOTIC2	FORTRAN logical tape unit, notice output reel 2, if any
6.	DATE	Date of computer run (optional)
7.	MIN	Minimum match percentage required to generate a notice on the basis of 'may' words
8.	NRAND	Maximum allowable number of random notices per user.

Other important primary program variables are:

1.	NOTES	Total number of notices written
2.	NOTESR	Number of random notices written
3.	KXTRA	1 if normal run, 2 if restart run
4.	KEYGO	Degree of match <0 match on 'not' term (no notice generated) =0 no match at all (no notice generated) 0<KEYGO<100 percentage match on 'may' terms (conditional notice generation) =100 match on single word user 'must' (KEYGO then reset to 300 and a notice generated) 100<KEYGO<300 match on user phrase (notice is generated)
5.	KORE	Length of array USER.

The subroutines used, in order of their logical occurrence within the primary program, are:

1.	ERASE	Calculates the size of array USER and stores the calculated value in cell KORE
2.	DATA	Initializes the reading of document and user profiles from tape
3.	MATCH	Compares the terms of a given pair of user and document profiles in order to determine the number of words found in both profiles i.e., the number of words that match. It returns a four-part answer:

MATCHS	Number of matching words used as single words
MATCHP	Number of matching words used in phrases
N	Number of matching words used both in phrases and as single words
KEYGO	>0 if match on a 'must' word <0 if match on a 'not' word

4. PHRASE — Attempts to match user phrases with document terms. With a non-zero input argument, it scans for 'not' phrases only. It can reset KEYGO if a 'must' or 'may' phrase matches (100<KEYGO<300) or if a 'not' phrase matches (KEYGO<0).

5. RANDM — Generates random numbers to determine if a random notice is produced at any given time. No user will receive more random notices than specified by the main program parameter NRAND, nor will he receive as random notices any he would (or would not) otherwise have received on the basis of his profile.

6. WRITE — Produces a notice upon command of the primary program.

7. NEXT — Secures a document-user pair of profiles for matching. It automatically controls all profile input after its initialization by subroutine DATA.

1.2 Input

1. Coded document profiles
2. Coded user profiles
3. Control card of program parameters.

1.3 Operating Instructions

The SDI MATCH program is a standard FMS job. The card deck for the system input tape includes in this order: ** job, * ID, *XEQ, source decks if any, binary decks if any, * DATA, control card, EOF. Tape units are assigned via the control card, one or two for user profiles, one or two for notices, and one for document profiles. All tapes must be mounted at the beginning of the job, and switching is then accomplished without operator intervention. A2 is expected as system input. EOJ occurs when matching is concluded or when an EOR is sensed on the

second reel of notice output. Exit is via the standard FMS routines, which are provided for this purpose as part of the MATCH I/O subroutines.

The MATCH program may be restarted at any point after profile matching has begun. To do so, obtain the profile numbers of the pair of profiles being matched (console display or memory dump), or of the last notice generated (last record on notice tape). Punch these numbers into the control card (see format), and begin the run again. The presence of the numbers will cause MATCH to scan to this pair on the user and document tapes, and then to resume matching at the next following pair.

1.4 Output

1. Document Notices

2. Record Formats

2.1 Input

1. Control Card

Cols. 4- 5	FORTRAN logical unit - document input
9-10	FORTRAN logical unit - user input, reel 1
14-15	FORTRAN logical unit - user input, reel 2, if any
19-20	FORTRAN logical unit - notice output, reel 1
24-25	FORTRAN logical unit - notice output, reel 2, if any
30-35	Date of computer run
39-40	Minimum match percentage required on 'may' notices
44-45	Maximum allowable number of random notices per user
50-55	User profile number if recovery run, otherwise blank
60-65	Document profile number if recovery run, otherwise blank.

For document and user profile formats, see the description of the VOCON program.

2.2 Output

1. Document Notices, unsorted, card-image records, 14 words per logical and physical record.

Col. 2	First user initial
3	Document type
4	Second user initial
5	Notice type Blank = may M = must

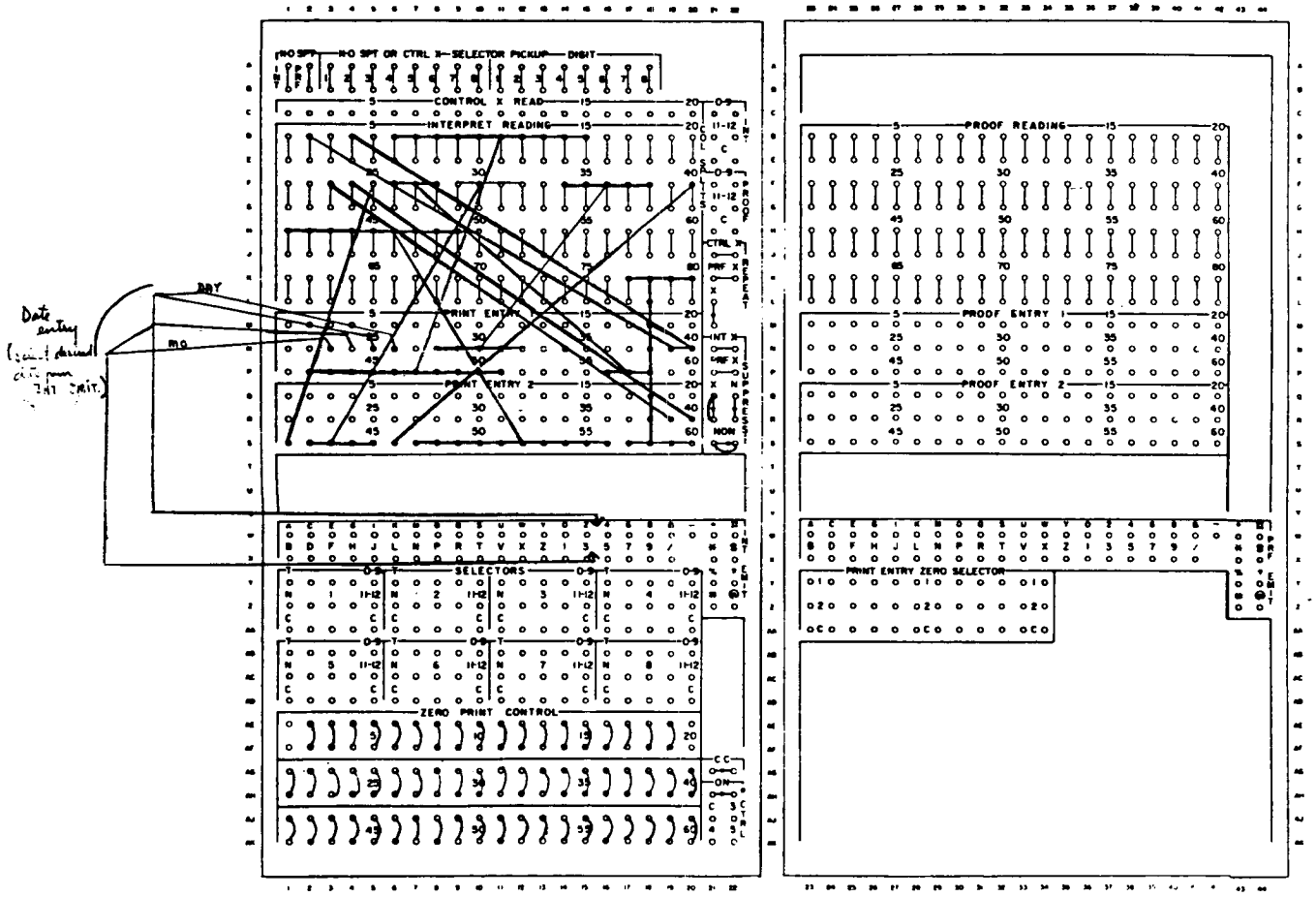
	P	= phrase
	9	= random
Cols.	6-15	User surname
	16-21	User profile number
	23-32	First ten characters of user address
	33-38	Document number
	40-50	Next eleven characters of user address
	77-80	Next four characters of user address

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

Printed in U.S.A.
Form 22-4850-1

557 ALPHABETIC INTERPRETER, CONTROL PANEL DIAGRAM



To be used for interpreting blue NASA cards. To print date, draw month and day from Int. Emit.

To interpret black cards from Black card program (1401), do not include the date. O/P from post cannot be interpreted with this board.

Print entry 1 prints on Line 1

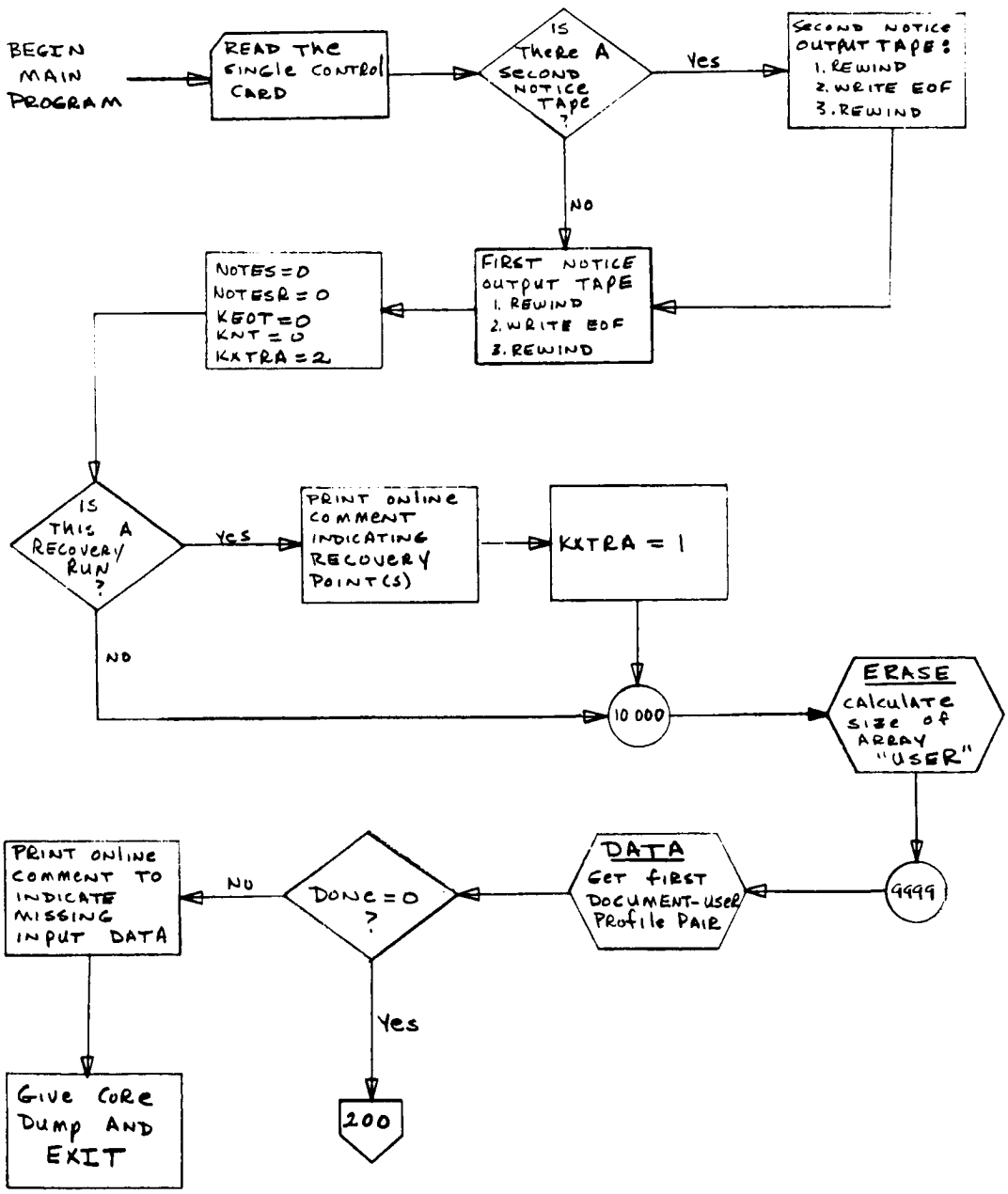
Print entry 2 prints on Line 1

NOTE:

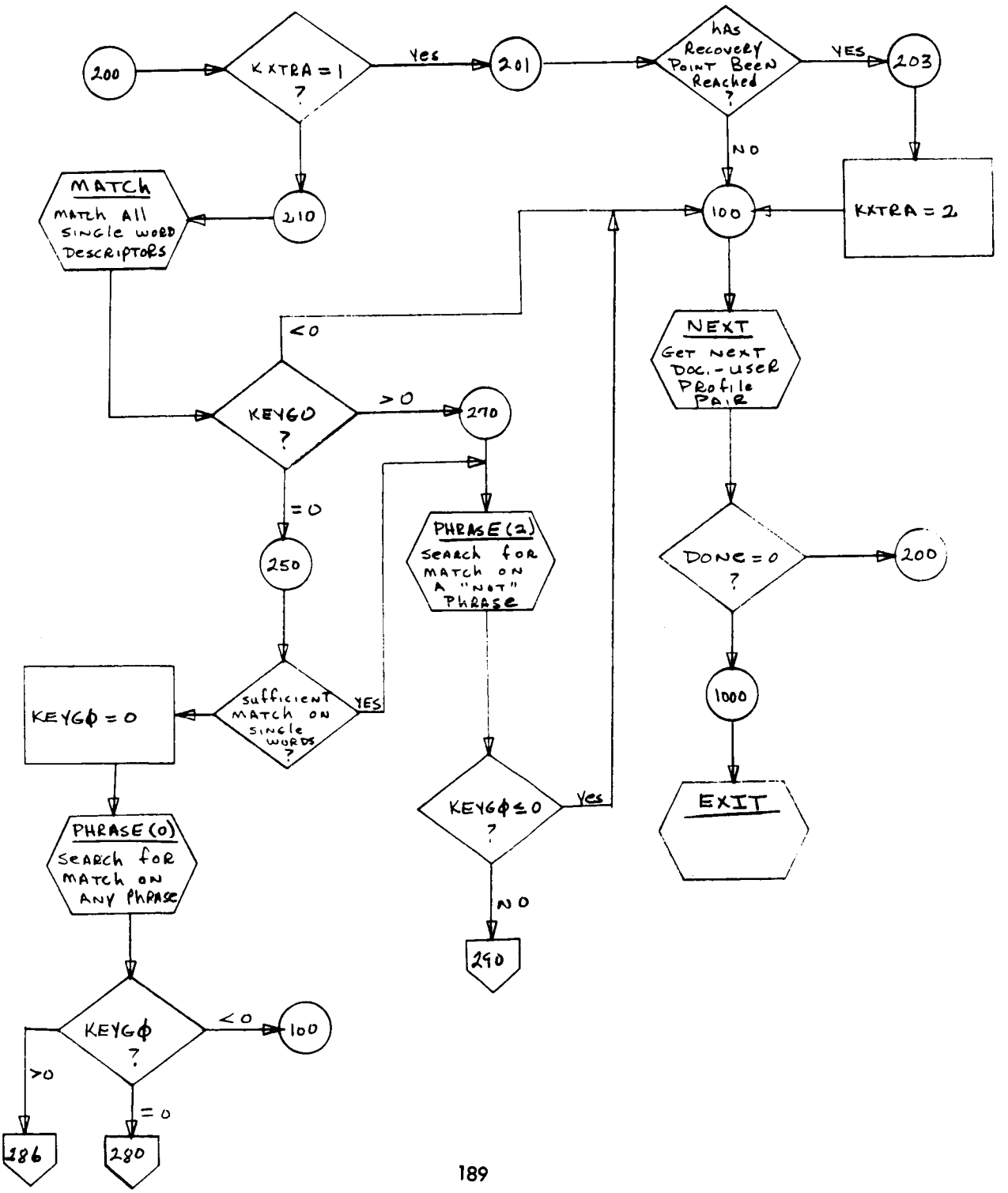
The original of this wiring diagram, with wiring shown in color, was supplied to NASA on September 28, 1964.

Figure 5. Control Panel Diagram for Interpreting
NASA-SDI Notices

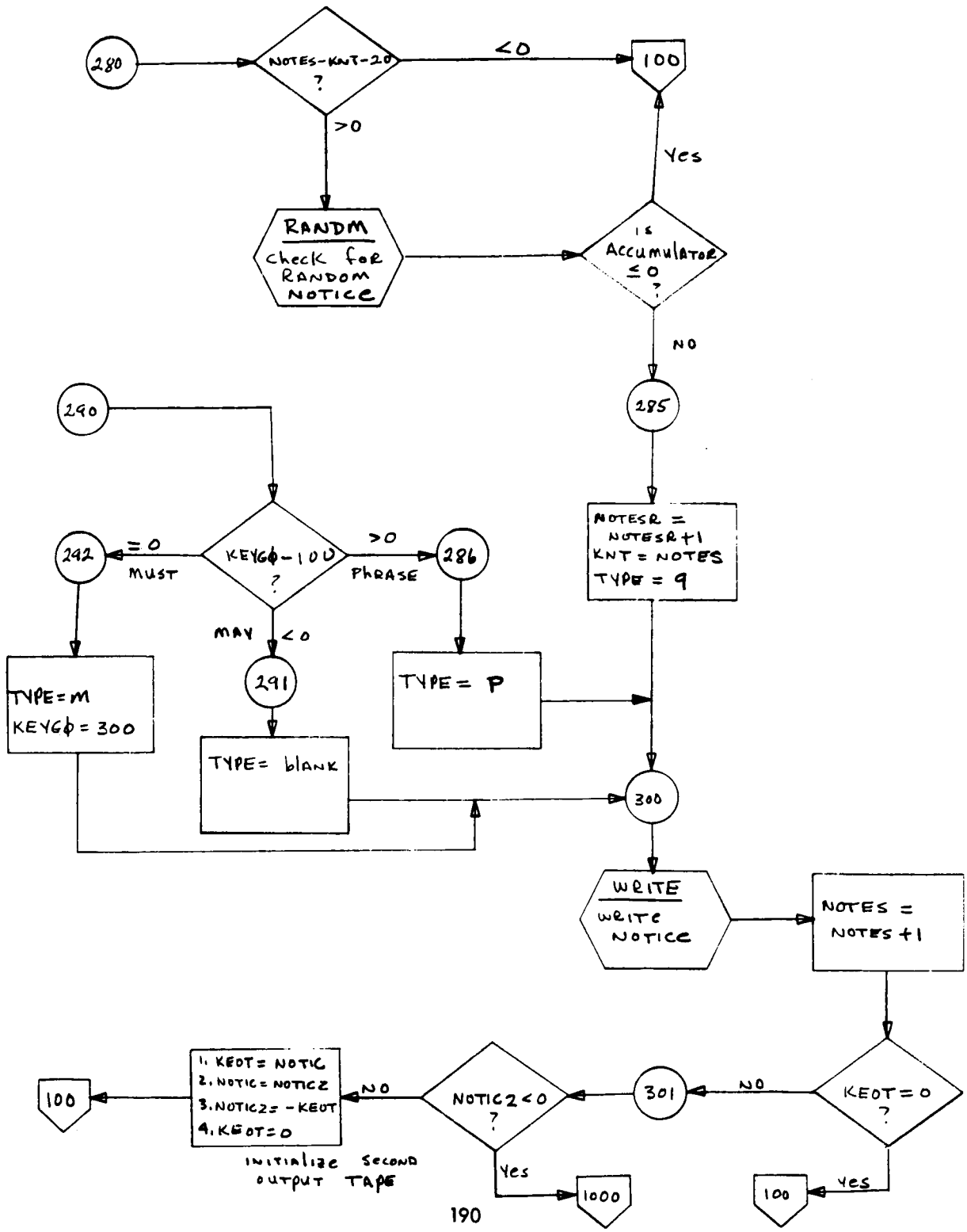
MAIN MATCH PROGRAM

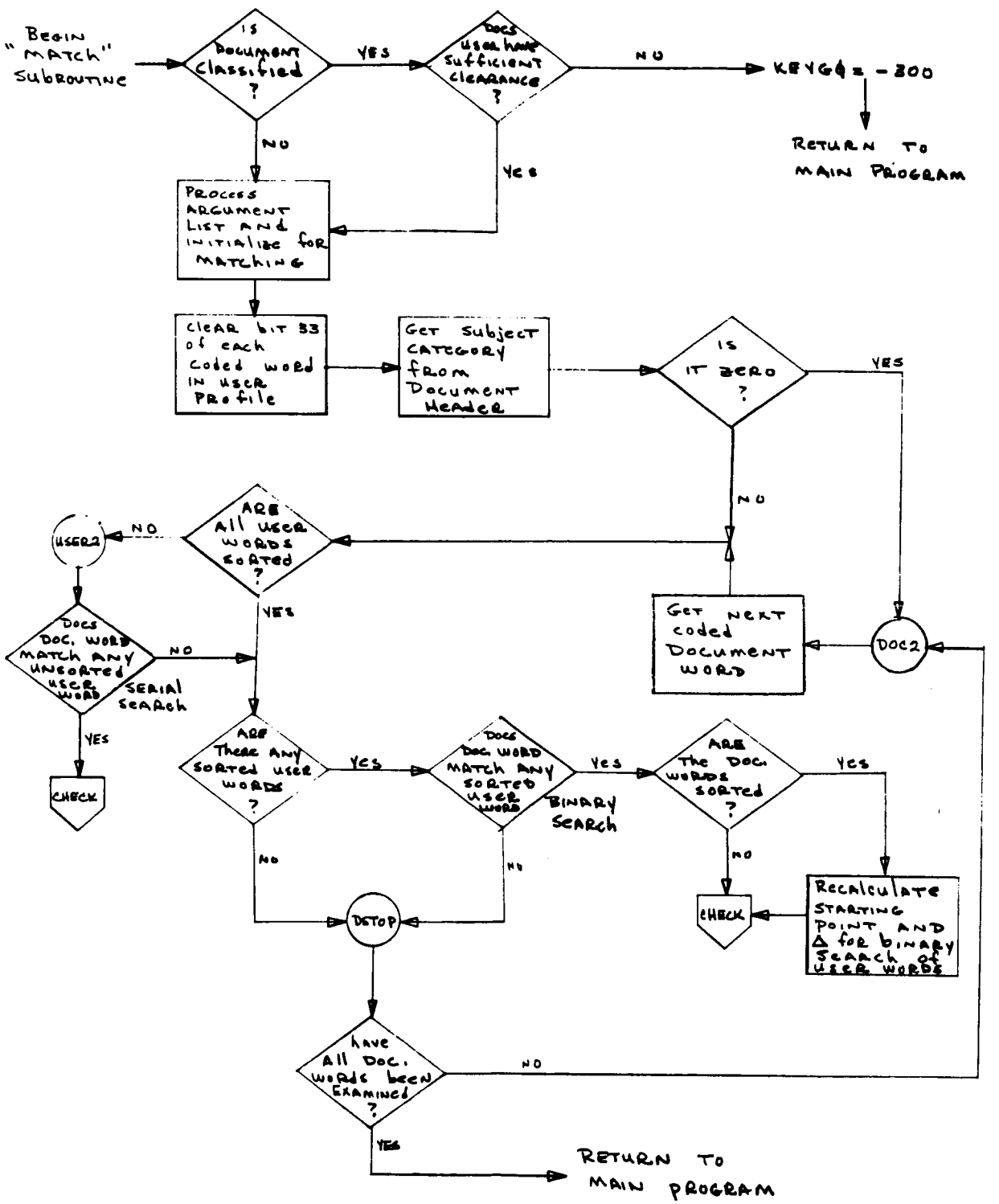


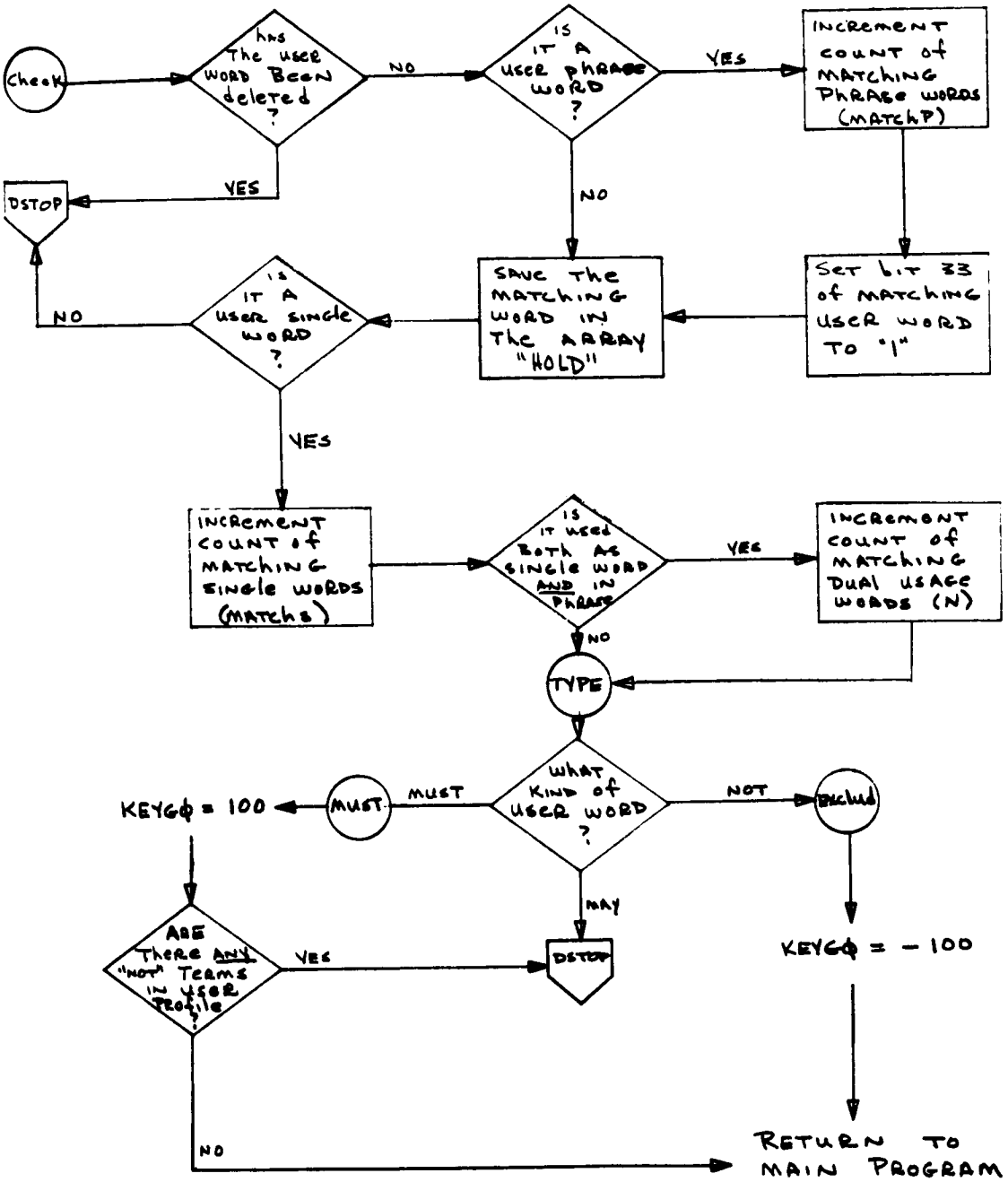
MAIN MATCH PROGRAM

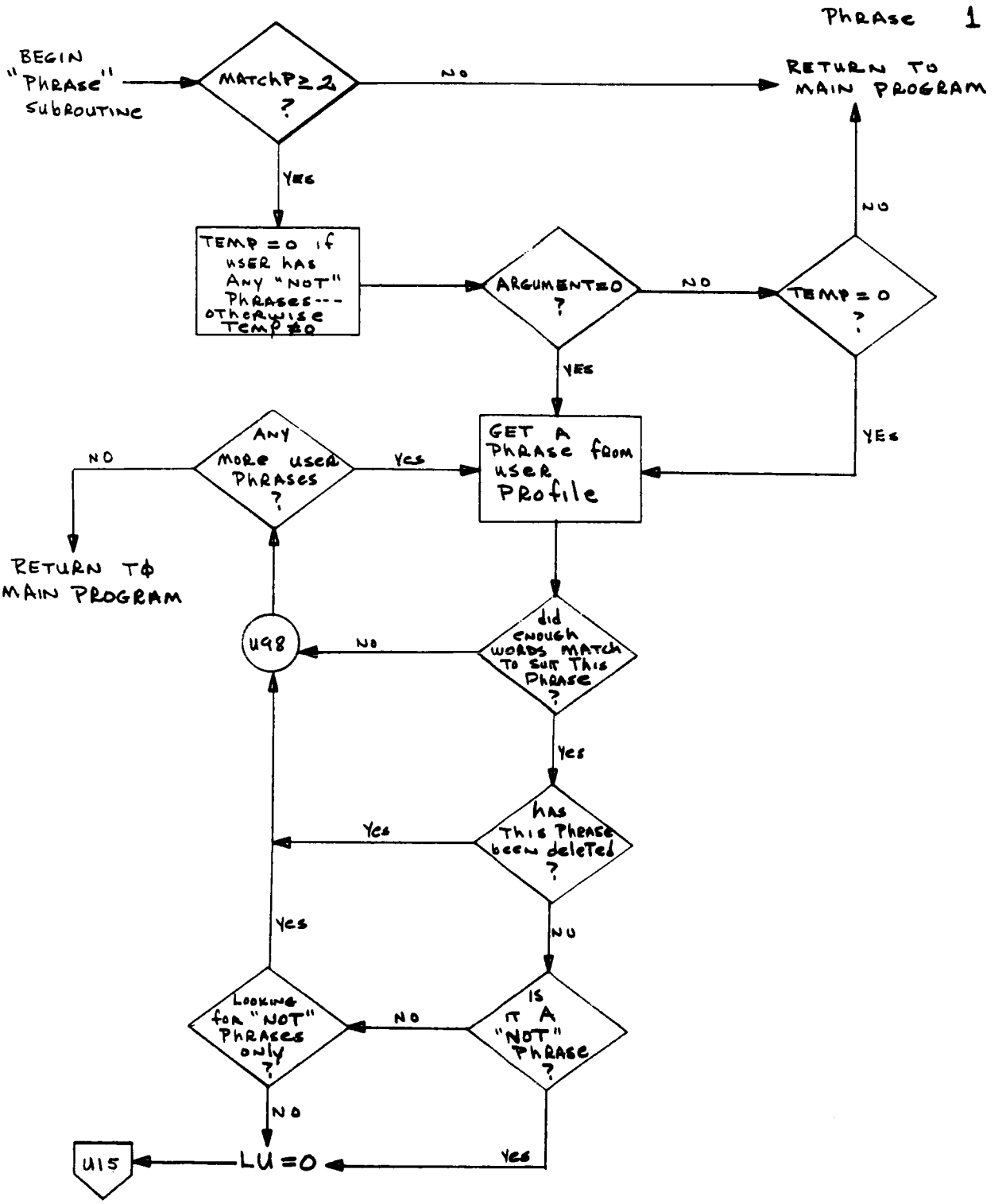


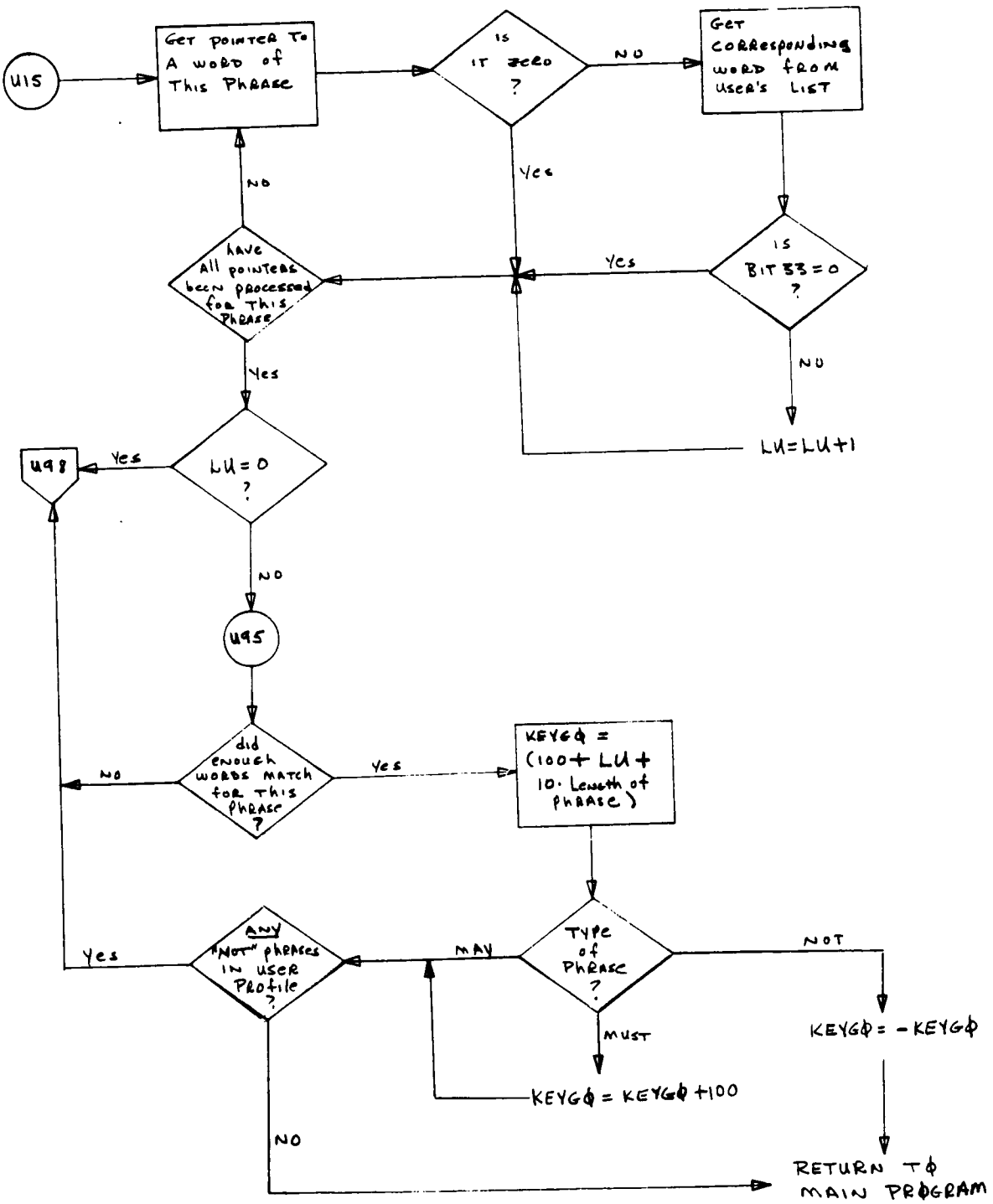
MAIN MATCH PROGRAM

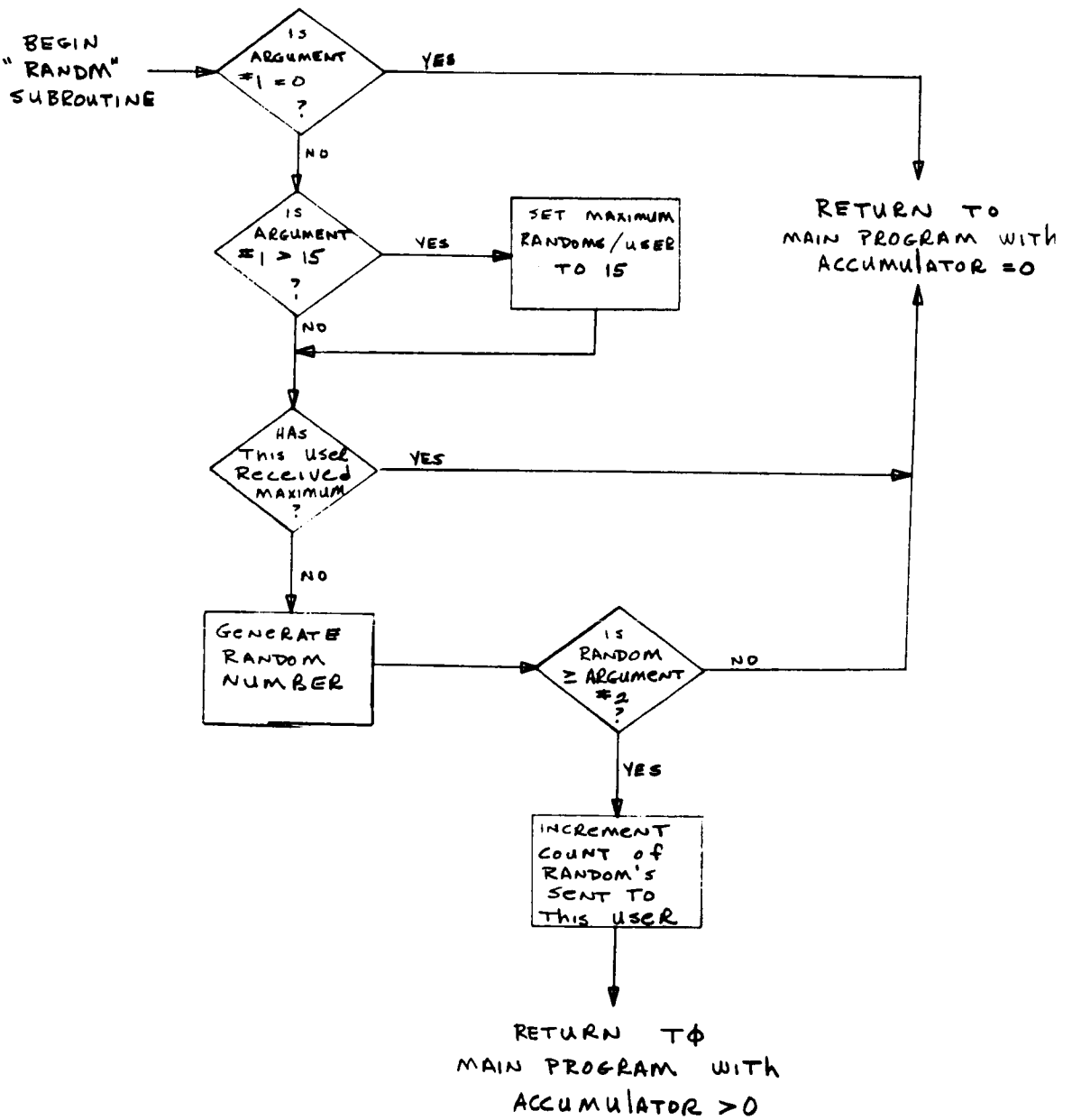



Begin "MATCH" Subroutine
Is Document Classified ?
YES
Does user have sufficient clearance ?
NO
KEYG@ = -300
Return to Main Program
NO
YES
Process Argument List and initialize for matching
Clear bit 33 of each coded word in user Profile
Get Subject Category from Document Header
Is it zero ?
YES
NO
Are All user words sorted ?
NO
USER2
YES
Does Doc. word match any unsorted user word
NO
Serial search
YES
Check
Are There any sorted user words ?
Yes
Does Doc. word match any sorted user word
Yes
Binary search
Are the Doc. words sorted ?
Yes
NO
NO
NO
DSTOP
Check
Recalculate starting point and Δ for binary search of user words
Get next coded Document word
DOC2
have All Doc. words been examined ?
NO
YES
Return to Main Program


Check
Has The user WORD Been deleted ?
NO
Is IT A USER PHRASE WORD ?
YES
INCREMENT COUNT of MATCHING PHRASE WORDS (MATCHP)
YES
DSTOP
NO
SAVE The MATCHING WORD IN The ARRAY "HOLD"
Set bit 33 of MATCHING USER WORD TO "1"
Is IT A USER single WORD ?
NO
YES
INCREMENT COUNT of MATCHING SINGLE WORDS (MATCHS)
Is IT used Both As SINGLE WORD AND IN PHRASE
YES
INCREMENT COUNT of MATCHING DUAL USAGE WORDS (N)
NO
TYPE
WHAT KIND of USER WORD ?
MUST
MUST
KEYGΦ = 100
NOT
Exclud
MAY
DSTOP
KEYGΦ = -100
ARE There ANY "NOT" Terms IN USER Profile ?
YES
NO
RETURN TO MAIN PROGRAM

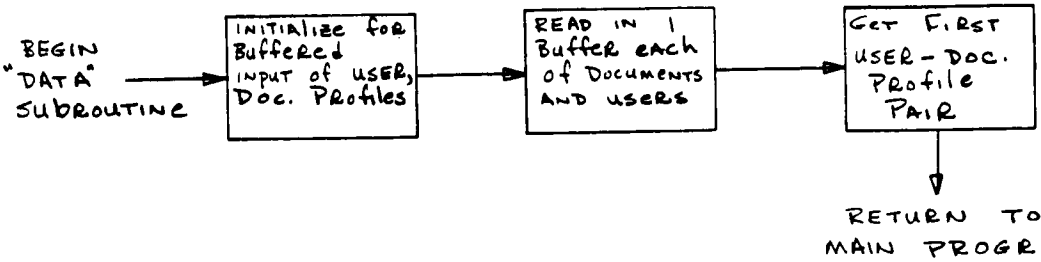

PHRASE 1
BEGIN "PHRASE" SUBROUTINE
MATCHP≥2 ?
NO
RETURN TO MAIN PROGRAM
YES
TEMP = 0 if USER has ANY "NOT" PHRASES--- OTHERWISE TEMP ≠0
ARGUMENT=0 ?
NO
TEMP = 0 ?
NO
YES
YES
GET A PHRASE from USER PROFILE
ANY MORE USER PHRASES ?
NO
YES
RETURN TO MAIN PROGRAM
U98
NO
did ENOUGH WORDS MATCH TO SUIT THIS PHRASE ?
YES
HAS THIS PHRASE been deleted ?
YES
NO
Looking for "NOT" PHRASES ONLY ?
NO
IS IT A "NOT" PHRASE ?
YES
NO
U15
LU=0

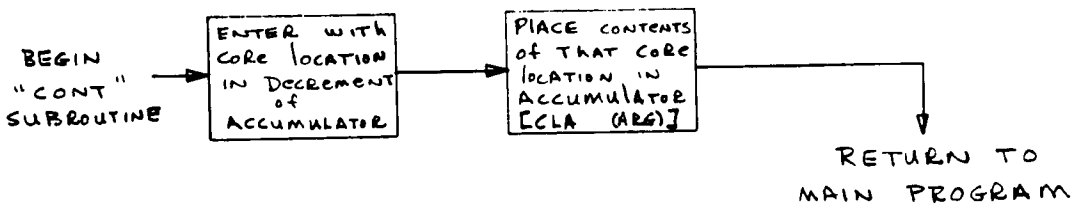

U15
GET POINTER TO A WORD of This PHRASE
IS IT ZERO ?
NO
GET CORRESPONDING WORD FROM USER'S LIST
Yes
NO
HAVE All POINTERS been PROCESSED for This PHRASE
Yes
IS BIT 53 = 0 ?
Yes
NO
LU=LU+1
Yes
U98
LU = 0 ?
NO
U95
NO
did ENOUGH WORDS MATCH for This PHRASE ?
Yes
KEYGΦ = (100 + LU + 10 · Length of PHRASE)
NO
Yes
ANY "NOT" PHRASES IN USER Profile ?
MAY
TYPE of PHRASE ?
NOT
MUST
KEYGΦ = KEYGΦ + 100
KEYGΦ = - KEYGΦ
NO
RETURN TΦ MAIN PROGRAM

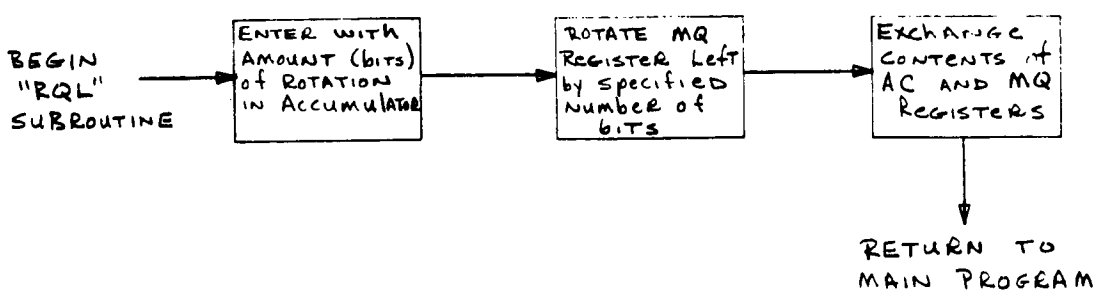

BEGIN "RANDM" SUBROUTINE
IS ARGUMENT #1 = 0 ?
YES
NO
RETURN TO MAIN PROGRAM WITH ACCUMULATOR =0
IS ARGUMENT #1 > 15 ?
YES
SET MAXIMUM RANDOMS/USER TO 15
NO
HAS This User Received MAXIMUM ?
YES
NO
GENERATE RANDOM NUMBER
IS RANDOM ≥ ARGUMENT #2 ?
NO
YES
INCREMENT COUNT of RANDOM's SENT TO This USER
RETURN TO MAIN PROGRAM WITH ACCUMULATOR >0

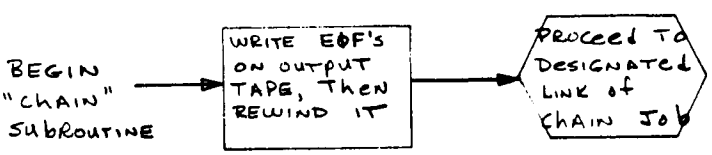
GENERAL IO PACKAGE

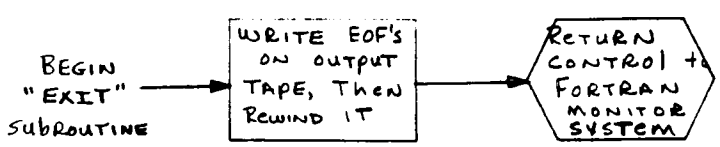
(DATA, CONT, RQL, CHAIN, EXIT)



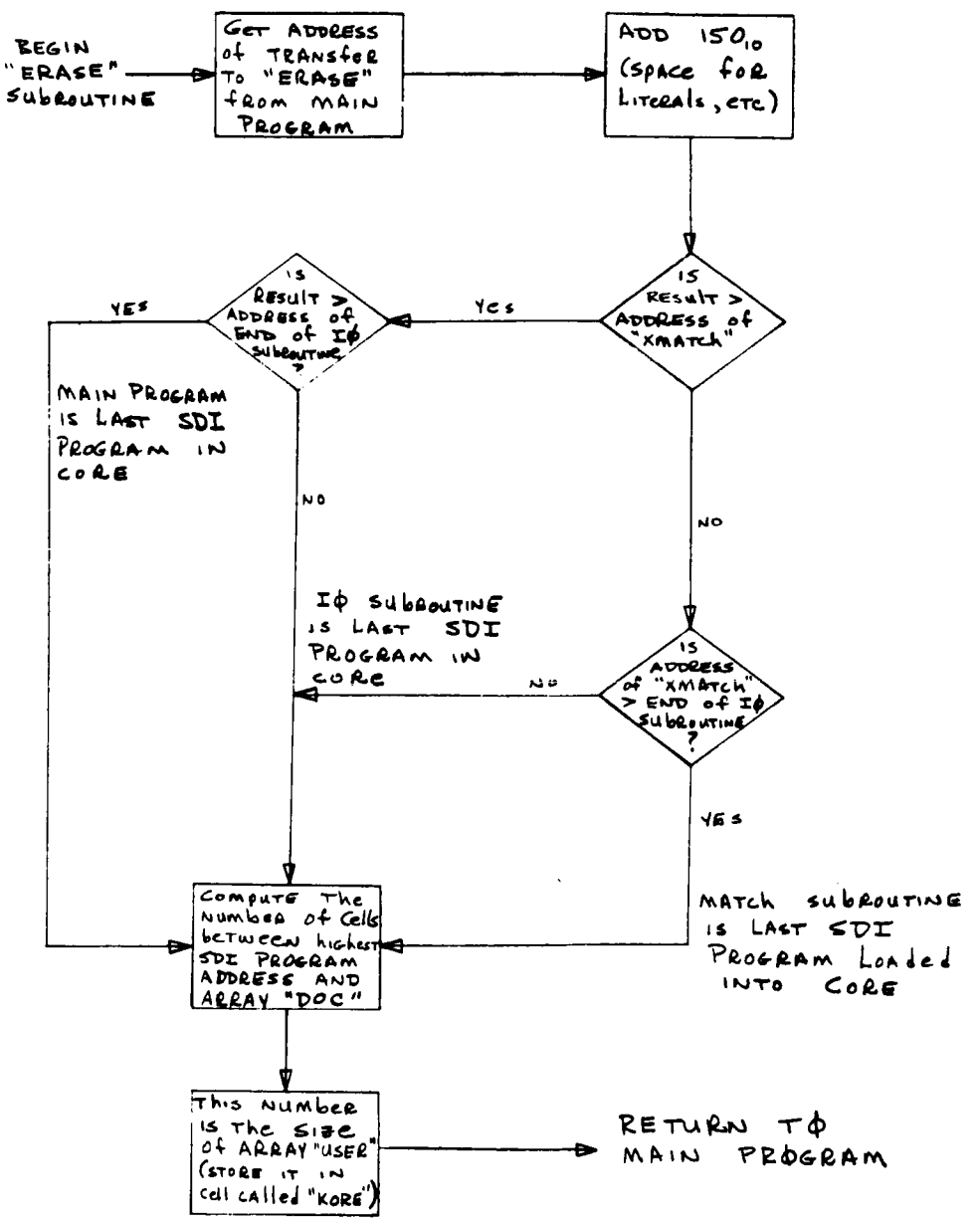




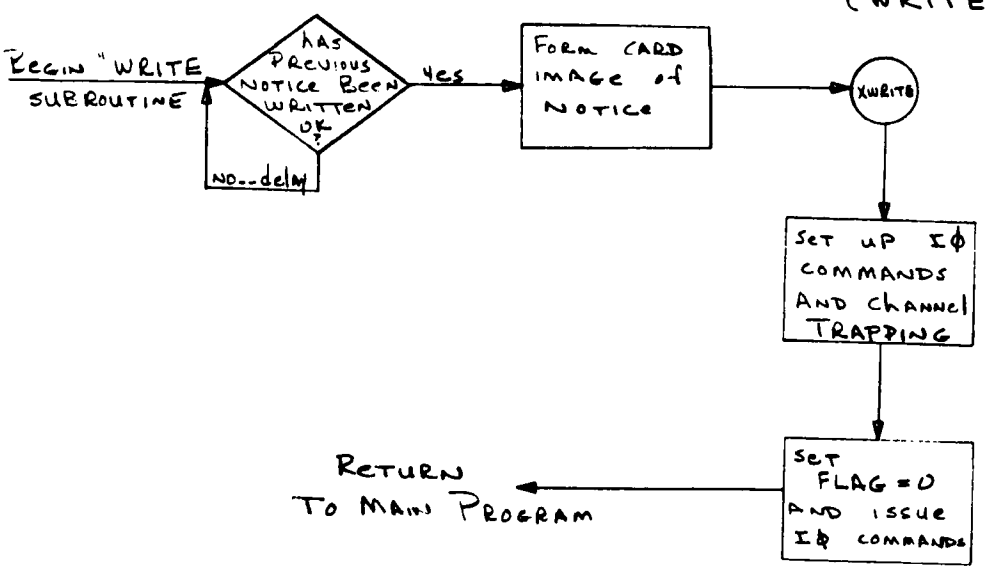


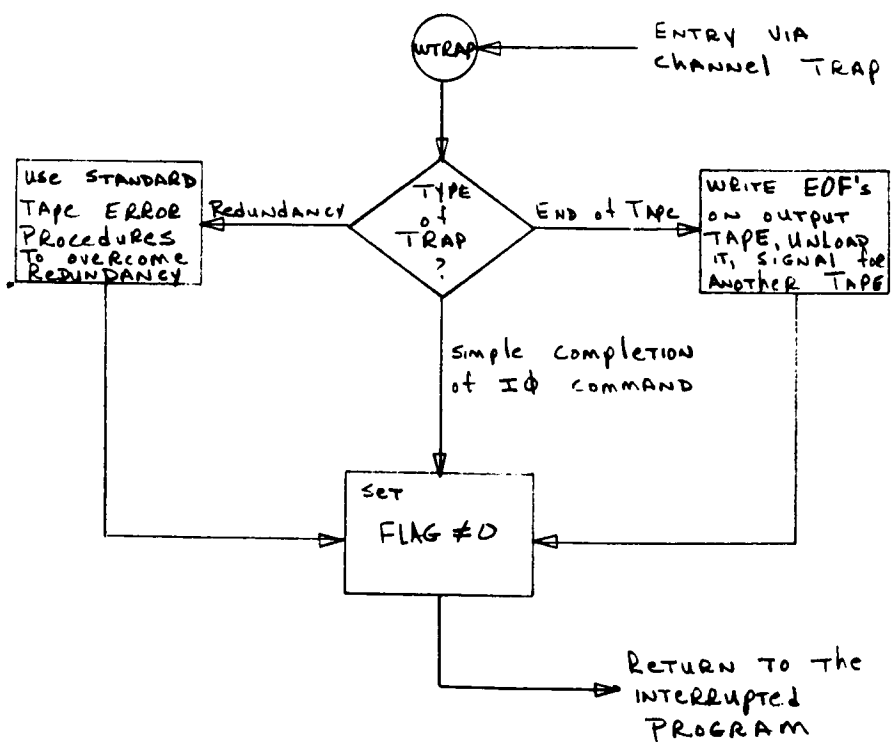


GENERAL IΦ PACKAGE
(ERASE)



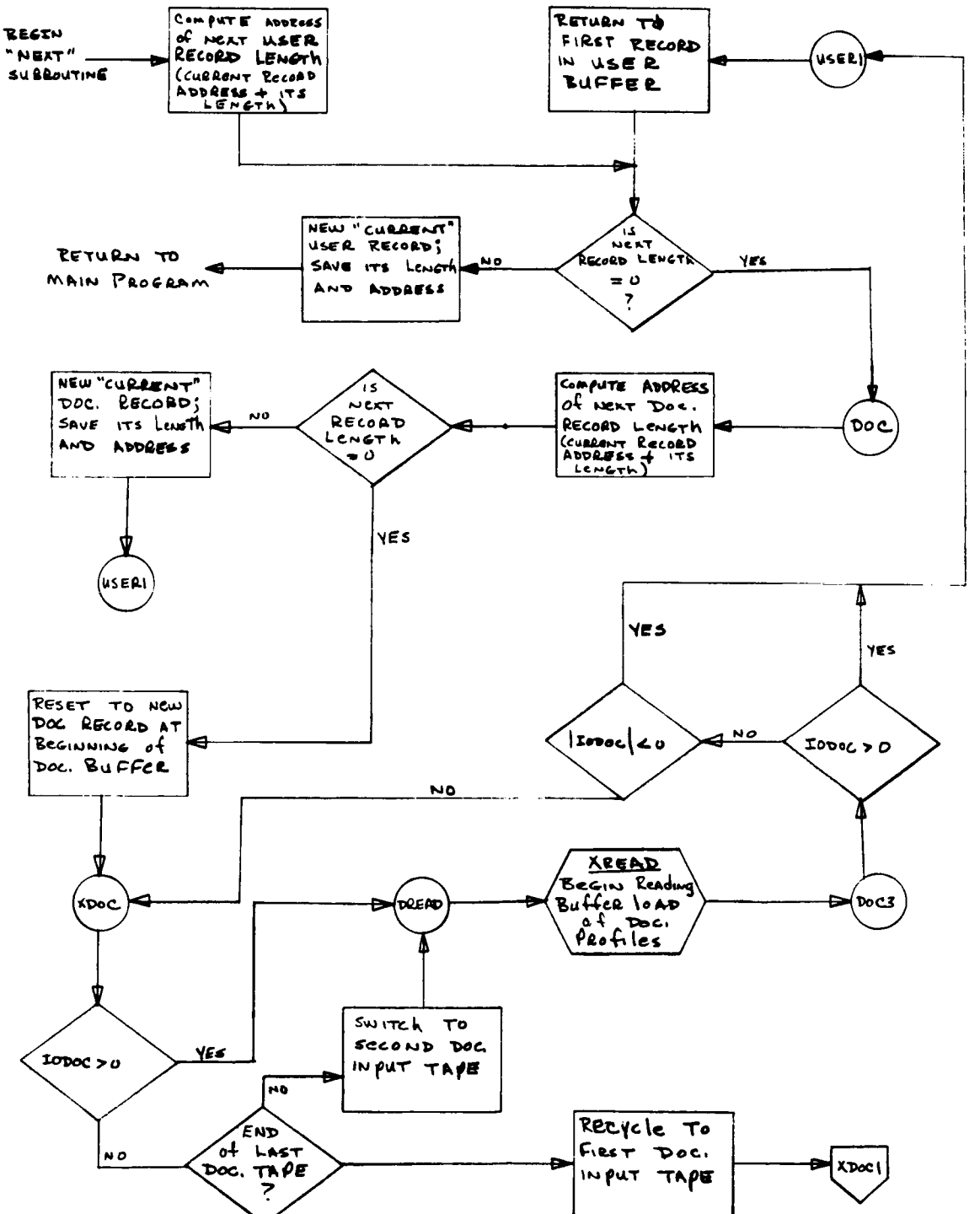
General I-ϕ Package (WRITE)





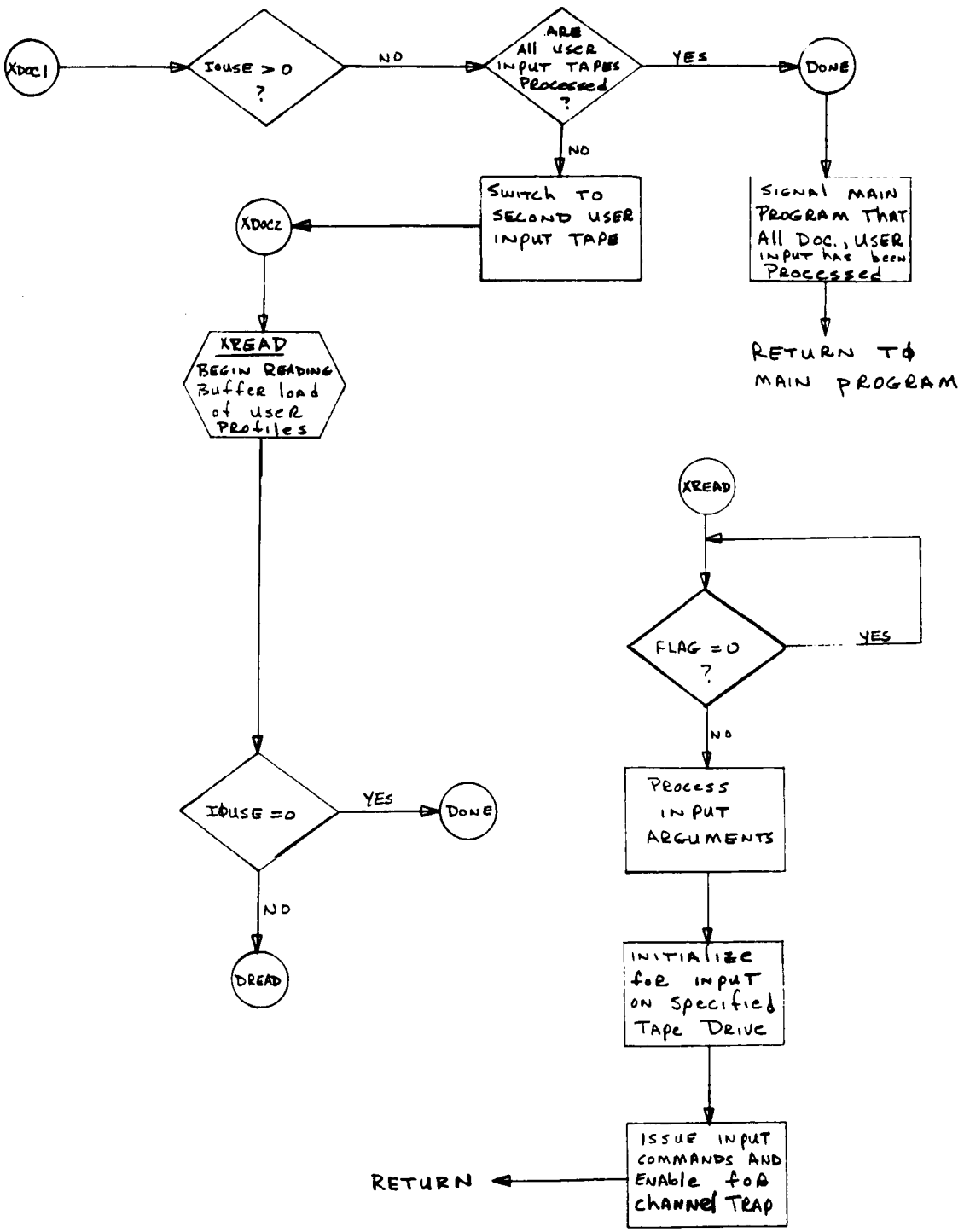
GENERAL IΦ PACKAGE

(NEXT 1)

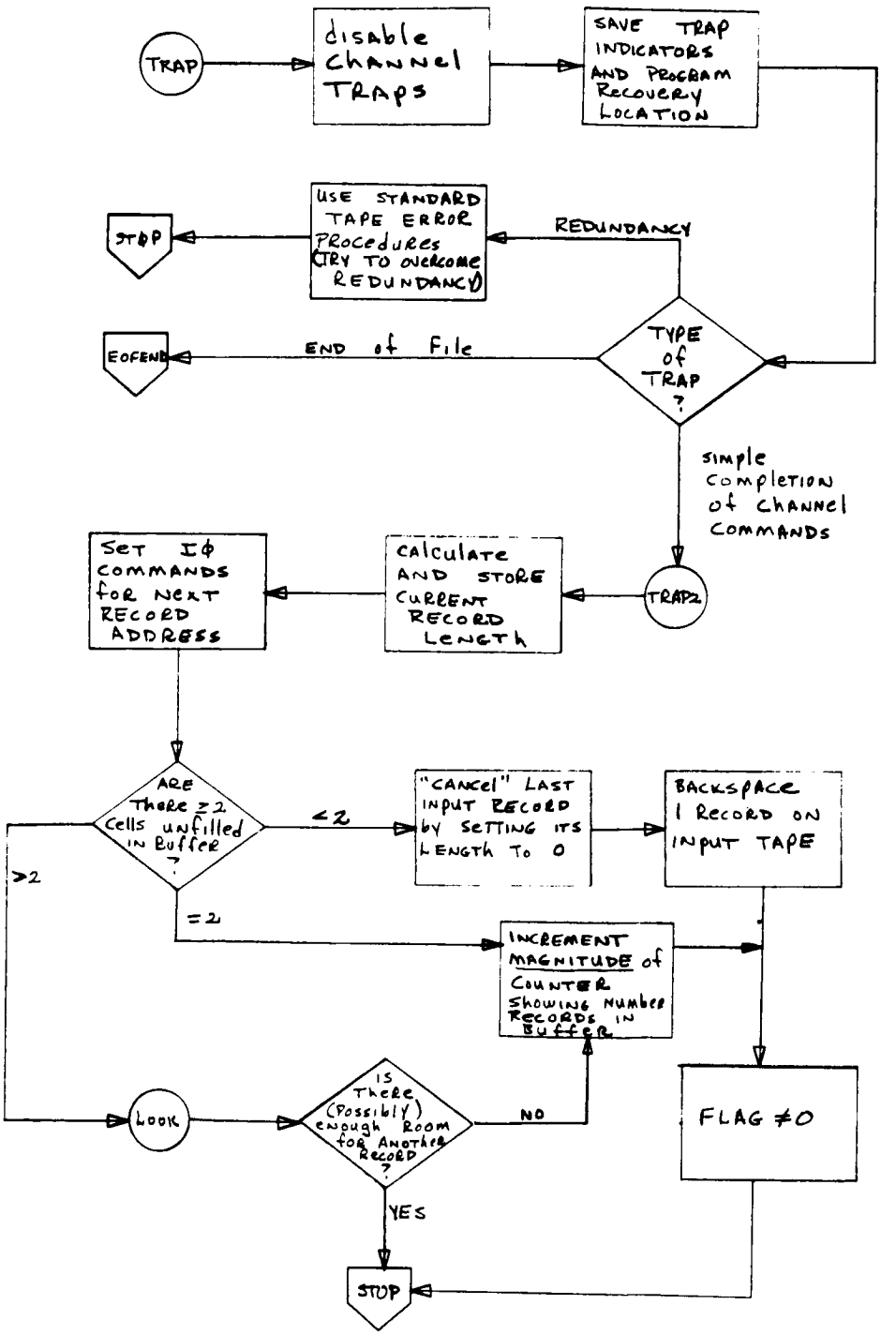


GENERAL IO PACKAGE

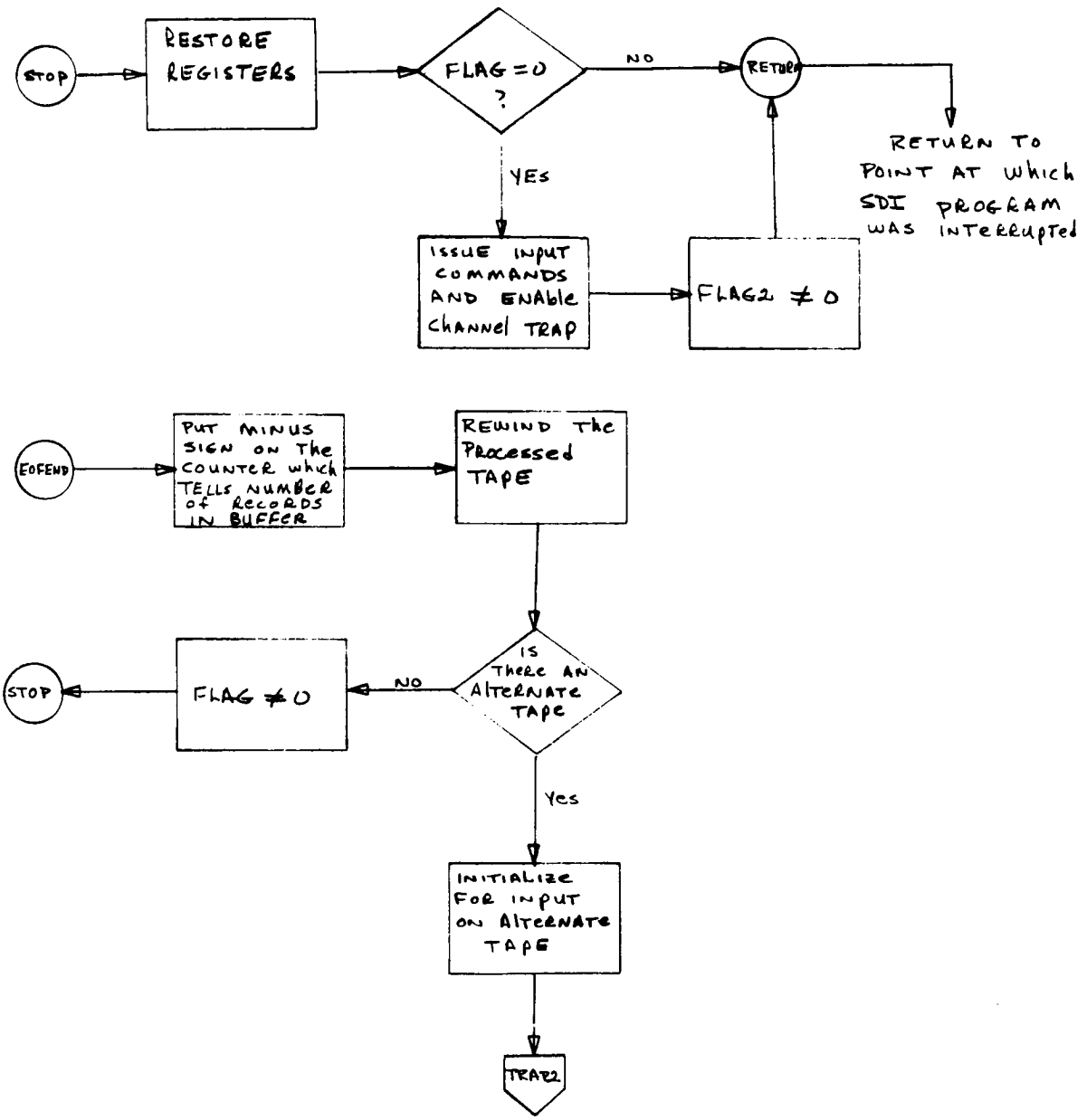
(NEXT 2)



GENERAL IΦ PACKAGE
(CHANNEL TRAP – INPUT)



GENERAL IO PACKAGE
(Channel Trap - Input)



D. SDI POST CARD-TO-TAPE PREPROCESSOR PROGRAM

1. Program Summary

1.1 Description

The SDI Post Card-to-Tape Preprocessor Program (NPOST), for the IBM 1401 Data Processing System, loads on tape user responses to SDI notices and POST program control cards so that the responses may be recorded in the historical notice response file by POST. The program requires a 1402 card read-punch, 1403 printer, two 729 or 7330 tape drives, and, optionally, a 1407 inquiry station; also, 8k core storage, a print line of 132 characters, high-low-equal compare and, optionally, sense switches. NPOST is written in 1401 Autocoder II.

As NPOST loads responses, it checks for validity, counts and analyzes them. It corrects responses where necessary and possible or rejects uncorrectable responses and separates those responses with a comments punch for manual handling. NPOST accepts as many as 6666 responses (the number POST can sort internally). After the last response is proeessed, NPOST prints the analysis and prepares a backup tape and a response listing.

Since a starting place should be supplied to POST for economical operation, NPOST provides the option either of having the program look for the five lowest document numbers and having the operator choose one of these as the starting place via the inquiry station, or of entering a starting place on a card among the responses.

1.2 Input

1. Card Read Feed: User response cards and POST control cards.
2. Card Punch Feed: Blank cards.

1.3 Operating Instructions

1. Ready the 1401 system: output tapes on units 2 and 3, printer with wide paper, card reader with condensed program deck followed by control and response cards, card punch with blank cards, check stop switch off, sense switch A on, sense switch F on if inquiry station option used.
2. Load program cards.
3. EOJ will be indicated on printer.
4. If sense switch F is on, additional instructions are printed at the inquiry station.

1.4 Output

1. Tape Unit 2: User responses and POST control cards.
2. Tape Unit 3: Duplicate of tape 2.
3. Printer: Analysis and listing of tape 2.
4. Stacker NR: Accepted responses with comments punches.
5. Stacker 1: Accepted responses.
6. Stacker 8/2: Rejected responses with uncorrectable errors.
7. Stacker NP: Blank cards (fed for timing).
8. Card Read Feed: Responses in excess of 6666.

2. Record Formats

2.1 Input

As user responses (Item 7) are returned to the SDI System, each day's accumulation is headed with a date card (Item 6) containing the date of receipt. About four boxes of such date-and-response-card sets are the maximum input to NPOST, headed by POST control cards (Items 1-5). The order of the cards is the order of the following list. For more information on the purposes of the POST control cards, see the description of the POST program.

Supply Items 1-5 as necessary. Only Item 1 is mandatory.

1. Second Notice Date (1 or more, 10 max.)
   Cols. 1 C
   41-44 Date, numeric month and day

2. User Closeout (15 max.), optional.
   Cols. 16-22 User number
   71 7

3. Report Limits (10 max.), optional
   Cols. 1- 6 Lower limit document number
   7-12 Upper limit document number
   71 9

4. POST Starting Place, optional
   Cols. 33-38 Document number
   71 8

5. Document Supply Cancellation (10 max.), optional
   Cols. 1- 6 Lower limit document number
   7-12 Upper limit document number
   71 8

Supply sets of Items 6 and 7 as explained above.

6. Response Receipt Date
   Cols. 1 * (asterisk)
   41-44 Date, numeric month and day

7. 1st and 2nd Responses (mixed)
   Cols. 16-22 User number
   33-38 Document number
   39 2 if second response
   71,73,75 Response code: +, -, 0, 2 are valid responses, 3 or T in 75 signifies a comment

   Other information punched in the response card is not utilized by NPOST or POST.

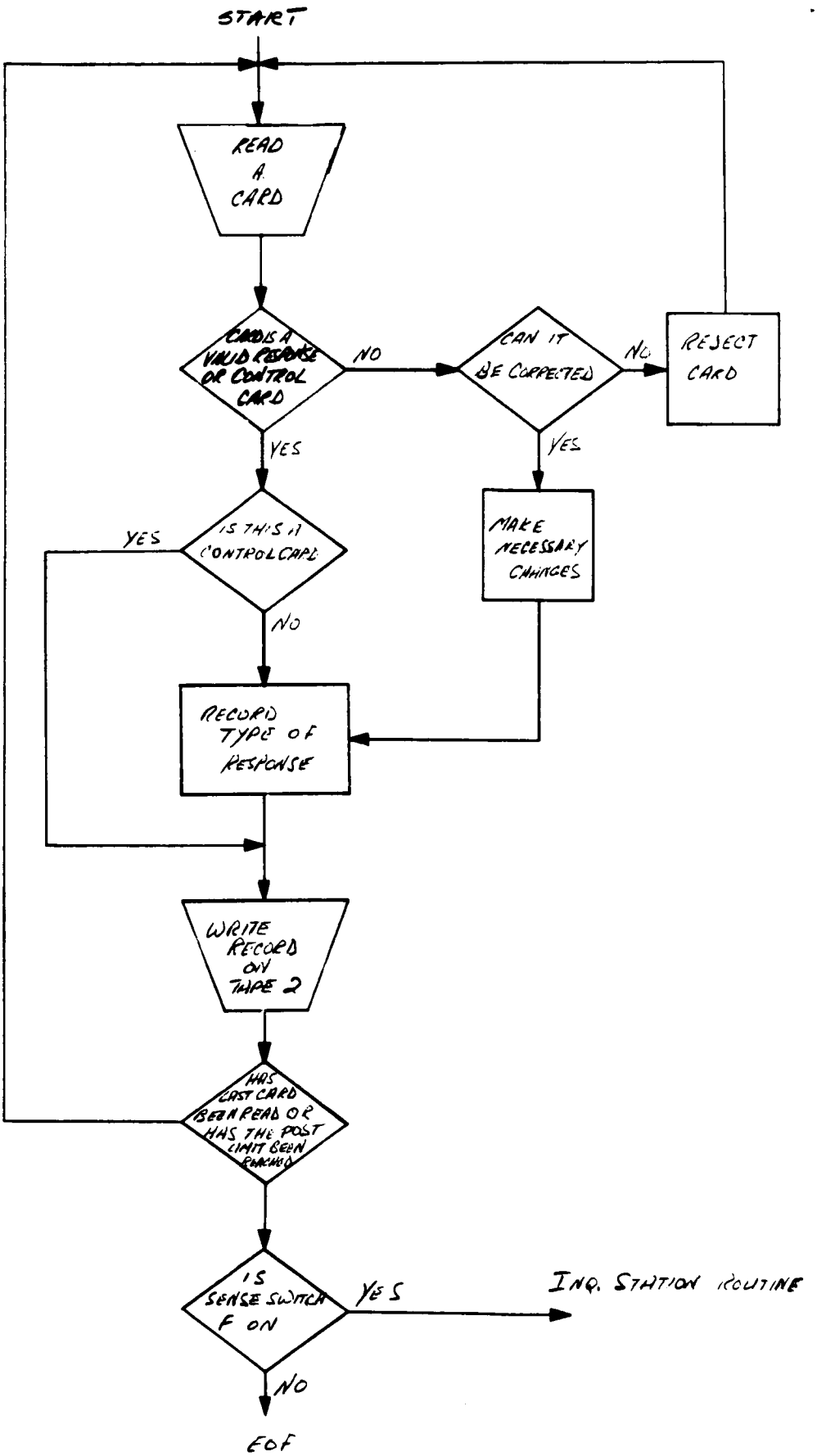
2.2 Output

1. Tape Unit 2: Card image of control card and response card input, suitable for BCD input to IBM 7090/94 Data Processing System, 84 characters or 14 words per physical and logical record, one file. Exceptions: column 1 of Items 1 and 6 is blank.

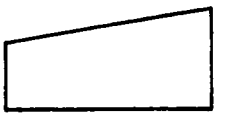
2. Printer: Analysis and listing.

3. Stacker 8/2: Sets of rejected responses, each set preceded by a duplicate receipt date card. The date card is punched whether or not any responses of that set are rejected.

The formats of outputs not listed above are either identical to inputs or are self-explanatory.


START
READ A CARD
CARD IS A VALID RESPONSE OR CONTROL CARD
NO
CAN IT BE CORRECTED
NO
REJECT CARD
YES
YES
IS THIS A CONTROL CARD
YES
MAKE NECESSARY CHANGES
NO
RECORD TYPE OF RESPONSE
WRITE RECORD ON TAPE 2
HAS LAST CARD BEEN READ OR HAS THE POST LIMIT BEEN REACHED
IS SENSE SWITCH F ON
YES
INQ. STATION ROUTINE
NO
EOF

INQUIRY STATION ROUTINE (OPTIONAL)



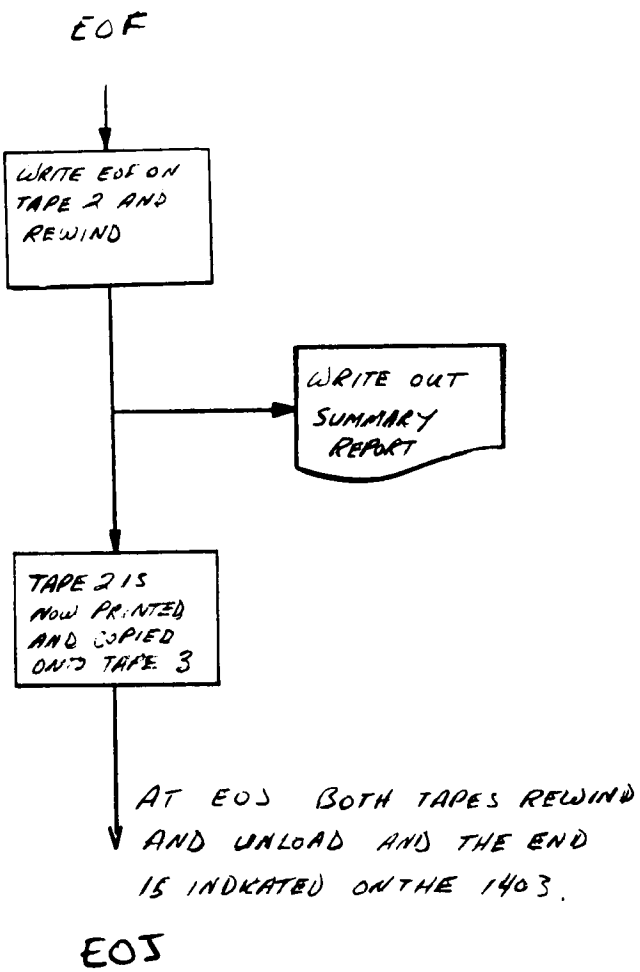
LIST FIVE (5) LOWEST DOCUMENT NUMBERS

ALLOW OPERATOR TO ENTER STARTING DOCUMENT NUMBER

IF THE OPERATOR IS SATISFIED WITH HIS CHOICE THEN WRITE THIS DOCUMENT NUMBER ON TAPE FOR POST CONTROL.

BRANCH TO EOF

[If desired the program will indicate the 5 lowest document numbers on the inquiry station and the operator may then indicate where he would want posting to begin. The operating instructions are expressed in the 1407 console printer.]


EOF
WRITE EOF ON TAPE 2 AND REWIND
WRITE OUT SUMMARY REPORT
TAPE 2 IS NOW PRINTED AND COPIED ONTO TAPE 3
AT EOJ BOTH TAPES REWIND AND UNLOAD AND THE END IS INDICATED ON THE 1403.
EOJ

E. SDI POST PROGRAM

1. Program Summary

1.1 Description

The SDI Post Program (POST), for the IBM 7090/94 Data Processing System maintains the historical notice-response file. The program runs under the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with eight 729 tape units (in addition to FORTRAN units). POST is written in FAP.

The historical notice-response file is produced by executions of POST and by the SDI MATCH program. It is maintained in document order. Within a document, notices are ordered by first response, second response and man number. Following the last notice of a given first-response code within a given document is a tally record showing the number of responses with that code, or in other words, the number of notices of the document preceding the tally record. The last tally record for a given document also includes the classification of the document, the current supply of copies and the total notices issued for the document.

The functions of POST in maintaining the historical notice-response file are:

1. Merging notices into the file.
2. Recording or posting user responses to the notices in the file and preparing second notices to accompany documents requested by users.
3. Modifying the file in the cases of users leaving the system and cancellations of document supplies.
4. Printing out portions of the historical file as requested.

The notices to be merged into the historical file are corrected records removed previously because of errors and additions of new material. The responses to be posted to the notices were loaded on tape by NPOST along with POST control cards. In phase 1, POST sorts the responses on document number and saves control information in appropriate tables. Responses for documents below the starting document number (starting-place control card), are dropped to be posted in a later run as are responses for documents not yet added to the historical file. POST then copies the historical file to the starting point, merging notice records as necessary. If any documents passes are within a report-limit range (report-limits control cards), they are written out on the report tape. In phase 2, response posting accompanies the copying and merging of the historical file and alternate notices. All records for a given document - historical notices, notices and responses - are loaded simultaneously in

core. If the document is within a cancellation-limit range (document-supply-cancellation control cards), the supply of copies is set to zero. Then responses and receipt dates are posted (response-receipt-date control cards). For every copy order a dated second notice is prepared if a copy is available (second-notice-date control cards); if not, a memo is prepared. When all responses for that document are posted, the program marks closeouts which are outstanding notices for users who have left the system (user-closeout control cards). Then, it sorts the notices, writes out the document if it is within a report-limits range and writes it out. As each document is processed, statistics are updated to summarize the activity in this run. The final statistics are then written out in a statement at the end of the run.

The tape unit assignments for POST files are set by EQU statements at the beginning of the POST program. They are:

1.	N	Input historical file, channel A, unit 5.
2.	J	Alternate notice file, channel A, unit 6.
3.	M	Response file, channel B, unit 5.
4.	I	Output historical file, channel B, unit 6.
5.	T	Report file, channel A, unit 8.
6.	H	Punch files - second notices, memos, exceptions - channel A, unit 7.
7.	G	Intermediate memo file, channel A, unit 4; copied to Item 6 at EOJ.
8.	F	Intermediate exception file, channel B, unit 1; copied to Item 6 at EOJ.
9.	W	Statement, channel A, unit 3; the system output tape.

Other important POST program variables include:

1.	CHANS	The number of channels used, currently 2.
2.	SIZE	Maximum number of notices per document, currently 1000.
3.	REC	Maximum number of logical records per physical record (blocking) of the historical notice-response file, as output. This variable depends on the setting of sense switch 2 (see Section 1.3).
4.	MAX	MAX = 14 * (REC + 1).
5.	MAXO	MAXO = 14 * (REC).
6.	NRS	Maximum number of responses per execution, currently 6666. The value of (3 * NRS) cannot exceed the value of (20 * SIZE). POST performs an internal sort on the responses; 6666 is the

maximum number that can be sorted.

7. CSIZE — Maximum number of closeouts per execution, currently 15.

8. RSIZE — Maximum number of report intervals per execution, currently 10.

9. SSIZE — Maximum number of supply cancellation intervals per execution, currently 10.

10. RQSIZE — Number of second notices to receive the same date, currently 250. RQSIZE is a reflection of the number of copies that the SDI clerks can mail in one day.

11. LPATCH — The address of the first unused location in the area PATCH. Because POST uses all space above itself for data, an area called PATCH is reserved for changes. LPATCH must be updated as the program is patched.

1.2 Input

1. Historical notice-response file.
2. Alternate notice file (for merging).
3. Responses and control cards (from NPOST).

1.3 Operating Instructions

The SDI POST program is a standard FMS job. The card deck for the system includes in this order: ** job, * ID, * XEQ, source decks if any, binary decks if any, EOF. A2 is expected as system input; A3 as system output; POST requires the I/O table from VOCON.

Sense switch settings are:

1. SS2 — ON if historical notice-response file is to be written unblocked.
2. SS5 — ON if alternate notice file is present.
3. SS1 — Set ON when last reel of historical file is mounted; on throughout run if only one reel.
4. SS3 — Set ON when last reel of alternate notice file is mounted; on throughout run if only one reel; OFF if no alternate file.

Program stops are:

1. $(3363)_8$ — If EOR historical file input, mount next reel and press START.
   If EOF, set on SS1 and press START.

2. $(3372)_8$ If EOR alternate notice file, mount next reel and press START.
If EOF, set on SS3 and press START.

3. $(3530)_8$ EOT on output historical notices; mount next reel and press START.

4. $(2631)_8$ EOJ. POST destroys any part of the monitor above itself, and thus cannot exit through the monitor.

1.4 Output

1. Historical notice-response file (updated)
2. Second notices
3. Memos
4. Exceptions
5. Document reports
6. Statement.

2. Record Formats

2.1 Input

1. Historical Notice-Response File

Starred fields are added to the records by the POST program. The file is sorted on document number, first response, second response and man number. It may be blocked or unblocked; see variable REC.

1a. Notice Records

Char.	
2	First user initial
4	Second user initial
6-15	User surname
16-21	User profile number
33-36	*Date of first notice
37-40	*First response receipt date
41-44	*Date of second notice
45-48	*Second response receipt date
55-60	Document number
71	*Response code of first response
72	*Response code of second response
75	*Digit 3 if comments punch
80	Notice type
	Blank = may
	M = must
	P = phrase
	9 = random

The allowable response codes in ascending sort order are:

+	Of interest, copy requested
-	Of interest, copy not requested
0	Of interest, have seen before
2	Of no interest
7	User closed out without response
Blank	No response

1b. Tally Records

POST expects a final tally record for each document from the MATCH program; if one is not provided, the document is assumed unclassified and the copy supply unlimited. POST creates intermediate tally records containing the fields identified by +.

Char.	
1	Classification Blank = unclassified + = classified
+50-53	*Number of notices with given first-response code
+55-60	Document number
64-65	Copy supply
74-77	Total number of notices for the document.

2. Alternate Notice File

The format of this file is the same as that of Item 1, unblocked.

3. Responses and Control Cards

The formats of these cards are listed in the discussion of NPOST, the program that loads them on tape.

2.2 Output

1. Historical Notice-Response File (updated)

The format of this file is the same as that of Item 1, input.

2. Second Notices

The format of this file is the same as the format of the second response, listed in the discussion of NPOST, with the exception that the response field has not yet been punched by the user.

3. Memos

The format of this file is the same as the format of the second response, listed in the discussion of NPOST, with the exception of the response field and the second-response identification code.

4. Exceptions

This file includes all valid responses that do not fall within the document range processed by a given execution and any notices

rejected as being out of document order. These records have the formats of their respective files, Items 2 and 1, input.

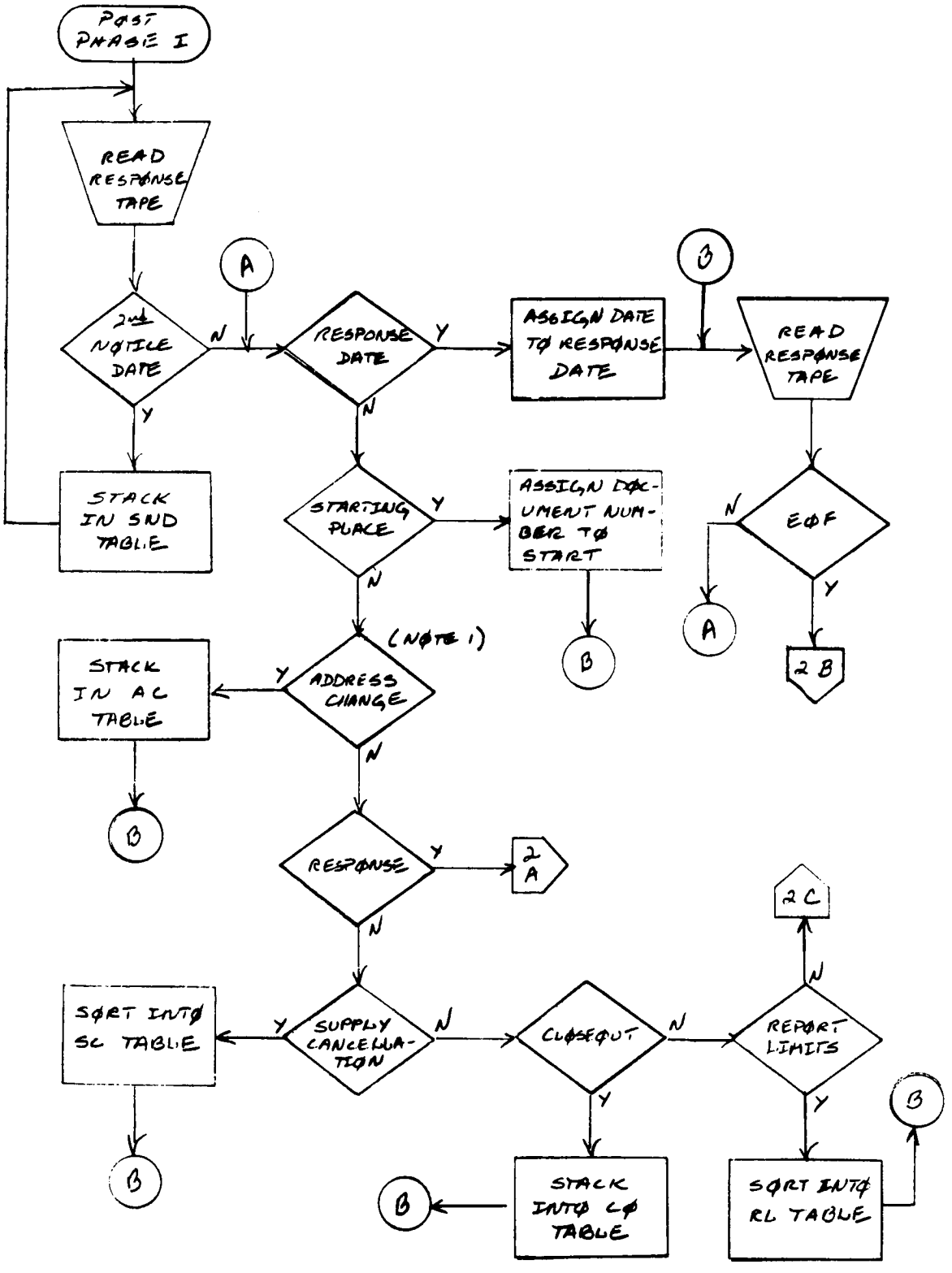
5. Document Reports

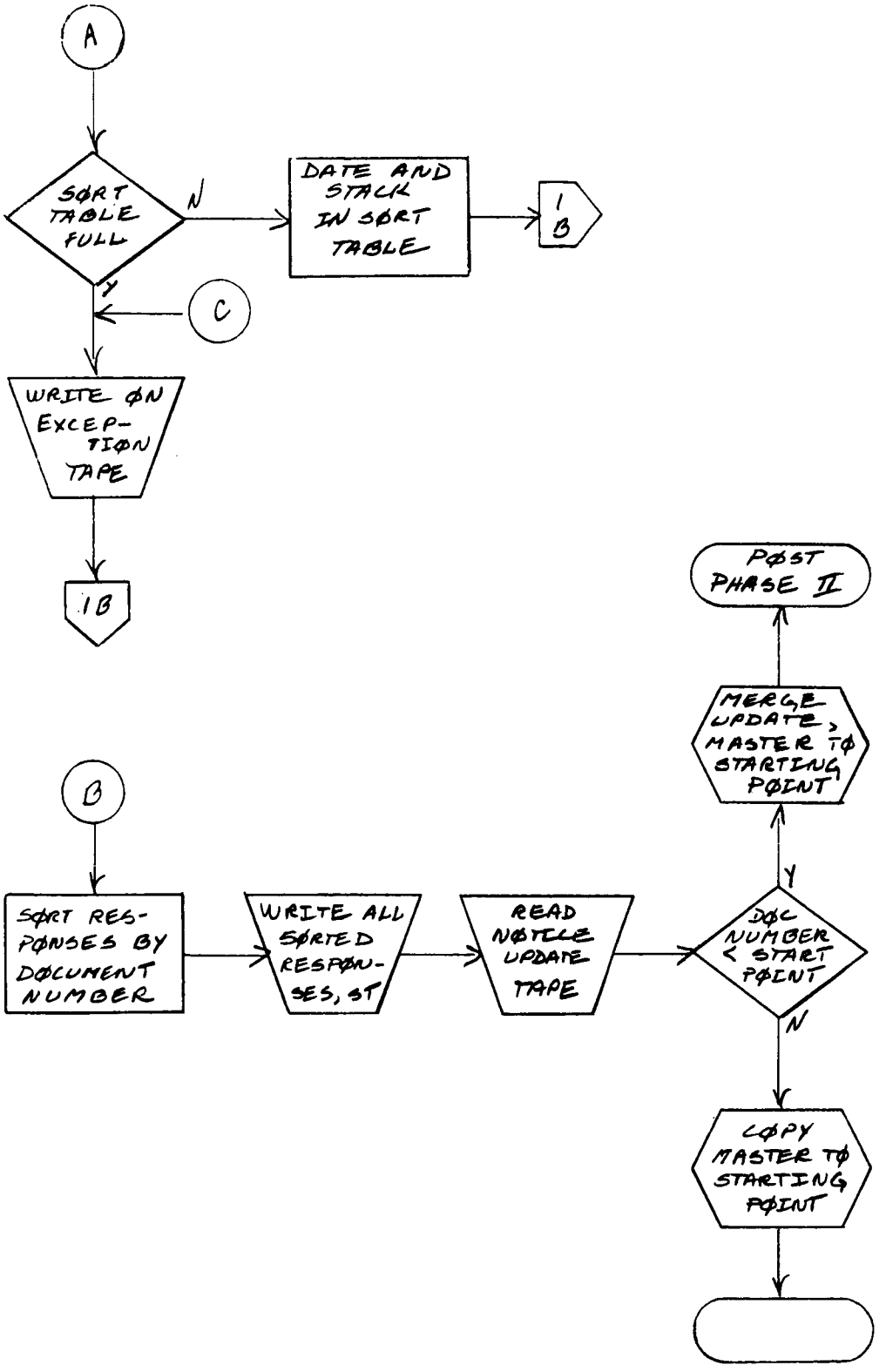
   The format of this file is the same as Item 1, input, unblocked.

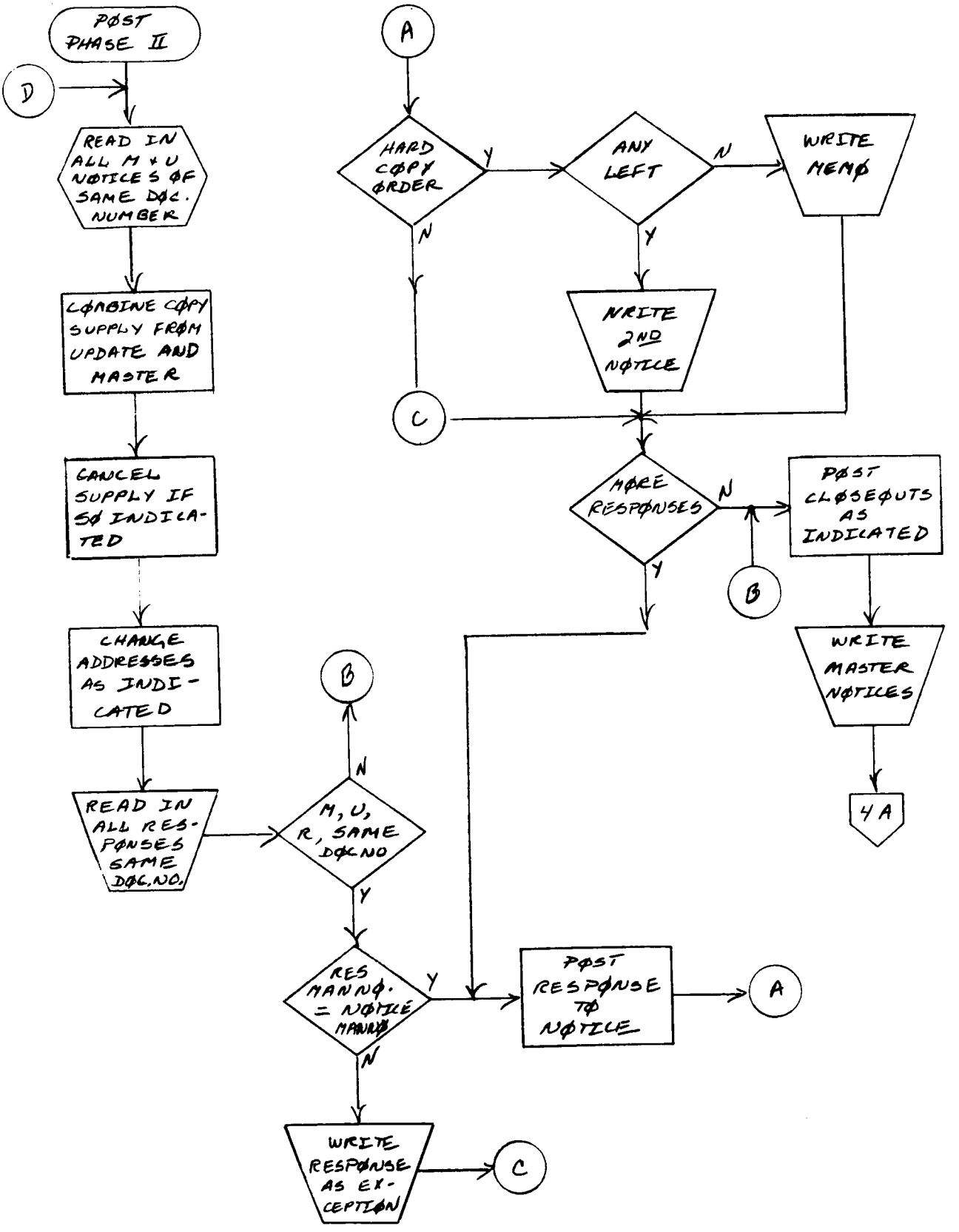
6. Statement

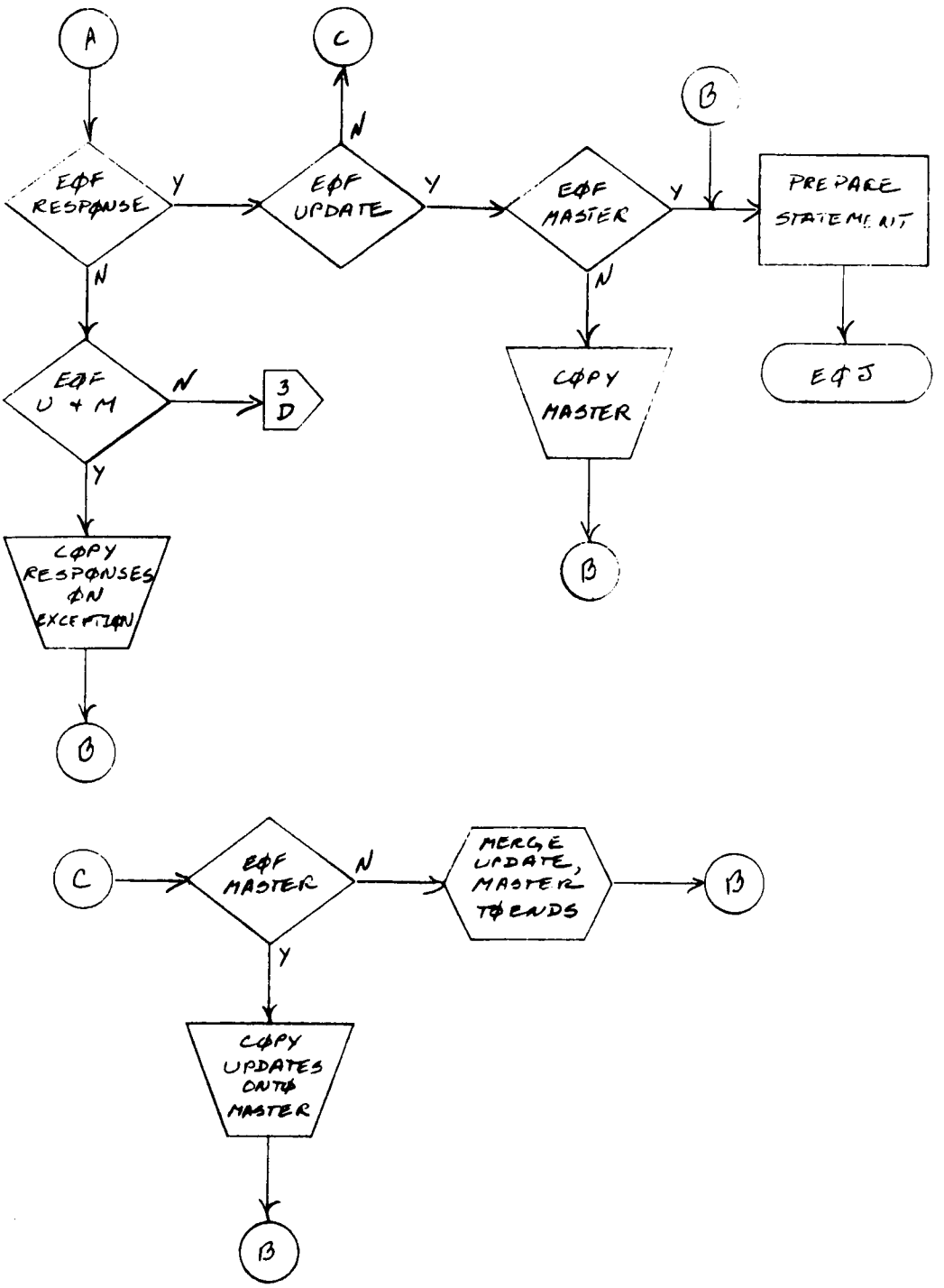
NOTE 1

The POST program was developed in another context. It is a useful adjunct to the NASA-SDI system, but as submitted, is not integrated completely into the system.


POST PHASE I
READ RESPONSE TAPE
A
2nd NOTICE DATE
N
Y
RESPONSE DATE
Y
N
ASSIGN DATE TO RESPONSE DATE
B
READ RESPONSE TAPE
STACK IN SND TABLE
STARTING PLACE
Y
N
ASSIGN DOCUMENT NUMBER TO START
B
N
EOF
Y
A
2B
(NOTE 1)
STACK IN AC TABLE
Y
ADDRESS CHANGE
N
B
RESPONSE
Y
2A
N
2C
N
SORT INTO SC TABLE
Y
SUPPLY CANCELLATION
N
CLOSEOUT
N
REPORT LIMITS
Y
B
Y
B
STACK INTO CO TABLE
SORT INTO RL TABLE


A
SORT TABLE FULL
N
DATE AND STACK IN SORT TABLE
1 B
Y
C
WRITE ON EXCEP-TION TAPE
1B
B
SORT RES-PONSES BY DOCUMENT NUMBER
WRITE ALL SORTED RESPON-SES, ST
READ NOTICE UPDATE TAPE
DOC NUMBER < START POINT
Y
N
MERGE UPDATE, MASTER TO STARTING POINT
POST PHASE II
COPY MASTER TO STARTING POINT


POST PHASE II
D
READ IN ALL M + U NOTICES OF SAME DOC. NUMBER
COMBINE COPY SUPPLY FROM UPDATE AND MASTER
CANCEL SUPPLY IF SO INDICATED
CHANGE ADDRESSES AS INDICATED
READ IN ALL RESPONSES SAME DOC.NO.
M, U, R, SAME DOC NO
N
Y
B
RES MANNO. = NOTICE MANNO
Y
N
WRITE RESPONSE AS EXCEPTION
C
POST RESPONSE TO NOTICE
A
A
HARD COPY ORDER
Y
N
ANY LEFT
N
Y
WRITE MEMO
WRITE 2ND NOTICE
C
MORE RESPONSES
N
Y
B
POST CLOSEOUTS AS INDICATED
WRITE MASTER NOTICES
4 A


A
C
B
EOF RESPONSE
Y
N
EOF UPDATE
Y
N
EOF MASTER
Y
N
PREPARE STATEMENT
EOJ
EOF U + M
N
3 D
Y
COPY RESPONSES ON EXCEPTION
B
COPY MASTER
B
C
EOF MASTER
N
MERGE UPDATE, MASTER TO ENDS
B
Y
COPY UPDATES ONTO MASTER
B

A. SYSTEM DESCRIPTION, NASA-SDI

SDI - the Selective Dissemination of Information - implements the philosophy that the supplying of relevant information to persons desiring it, from the bulk of information available, can be accomplished effectively and economically by machine.

SDI may be said to be the reverse of a conventional library operation. A library stores documents and waits for users to come in the door. An SDI system stores representations of users' interests, and, as documents come in the door, the system disseminates to users notices of documents that appear relevant to their interests.

An SDI user is represented in the system by a profile i.e., a list of descriptors that delineates his fields of interest. Information, the content of a document, is similarly represented. Periodically, the system edits user profile changes, new user profiles, and incoming document profiles against a vocabulary control guide, to insure that all profiles utilize the same descriptors to identify the same things. Then, the system compares these profiles with each other, selecting the documents apparently relevant to the stated interests of each user. For any comparison, matches on given numbers of descriptors determine relevancy. Notices of the documents are then sent to users. A notice consists of the document abstract and descriptors printed on a 3 x 5 card and of a Port-A-Punch® card with which copies of the document may be ordered and actual degree of relevance indicated.

Two major computer programs comprise the SDI system designed for the National Aeronautics and Space Administration, called NASA-SDI. These two programs are: the Vocabulary Control Program (VOCON), and the Profile Matching Program (MATCH). VOCON updates or originates the vocabulary control guide, updates or originates the user profiles, and edits document and user profiles against the vocabulary control guide. MATCH compares document and user profiles, generating document notices when the designated match criteria are met. In sections B and C following, VOCON and MATCH are presented from a programming and operating standpoint, including detailed record formats and program flow charts. They are shown schematically in Figures 1 and 2. For a condensed description of what is accomplished by VOCON and MATCH in terms of the SDI System phase, refer to Volume I of this report, Section B (NASA-CR-62020).

A third major computer program (POST), and its preprocessor (NPOST), are also presented in sections D and E following. They are shown schematically in Figure 3. POST is the SDI Posting Program. It maintains the historical notice-response file, recording or posting user responses to the notices in the file and preparing second notices to accompany documents requested by users. The purpose of the historical notice-response file is the accumulation of statistical information about system performance and the handling of document requests. NPOST loads on tape the user responses (returned notices) and POST control cards, checks the responses for validity, and limits the number of responses sent to POST at any one time.

CR-62021

POST and NPOST are not completely integrated into the NASA-SDI System. NPOST will load NASA-SDI responses on tape, and POST will post them to the historical notice-response file. But, the second notices prepared by POST would require further processing to add NASA addresses to them. Similarly, the notices generated by MATCH that form the skeleton of the historical notice-response file require a change in format before POST can use them. To circumvent the first discrepency, the NASA-SDI System utilizes second notices prepared ahead in quantity with the document number and date omitted. This information is added later when a specific document has been requested. To eliminate the second discrepency, IBM has supplied to NASA a 1401 program that changes the format of the MATCH notices for POST.

To summarize: Three programs are submitted to NASA and presented here as parts of an SDI System. Two of these, VOCON and MATCH, form the necessary parts of the NASA-SDI System. They were prepared by IBM to meet the specific needs of NASA. The third program, POST, was developed in another context. It is a useful adjunct to the NASA-SDI System, but, as submitted, is not integrated completely into the system.

CR-62021

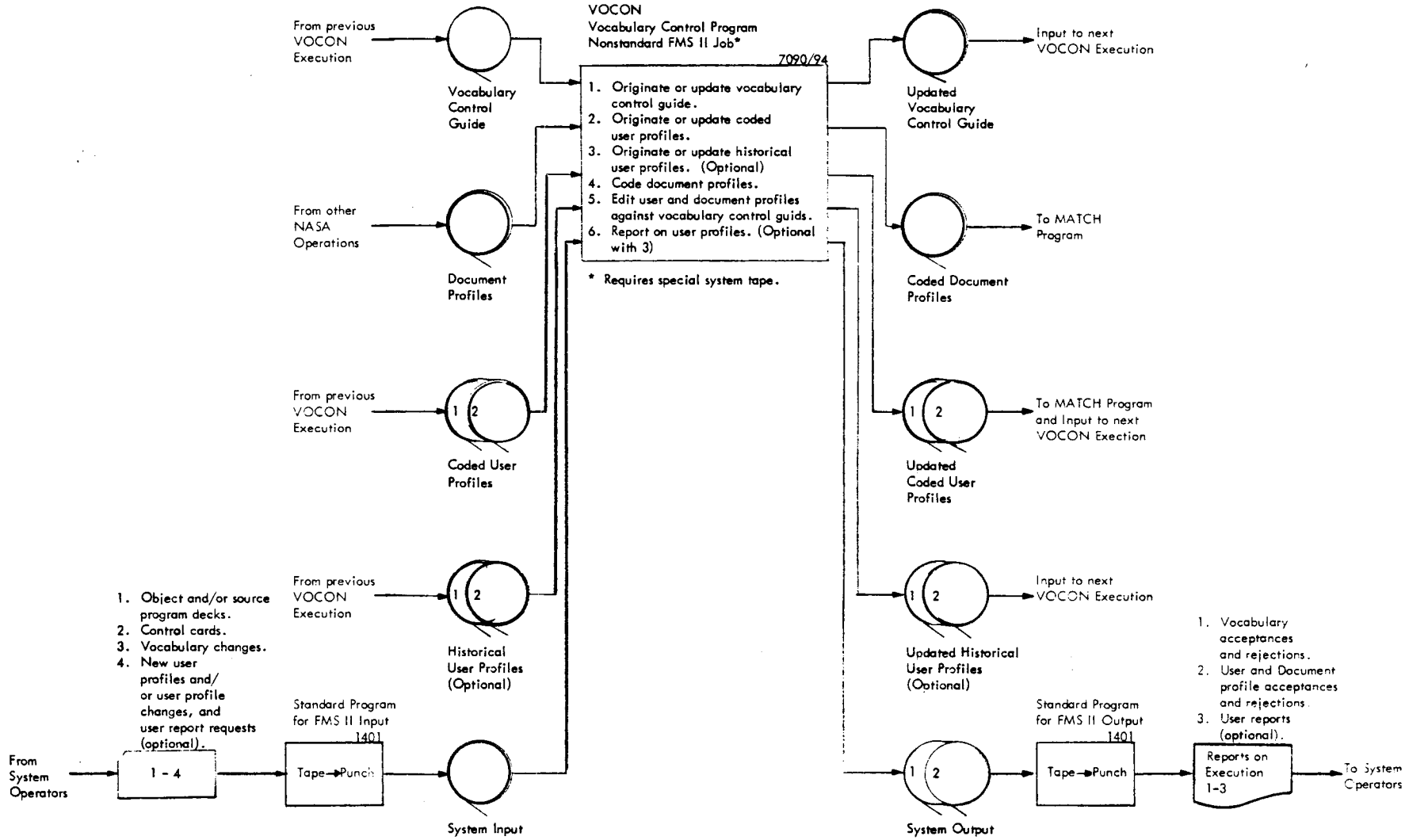


Figure 1. VOCON

CR-62021

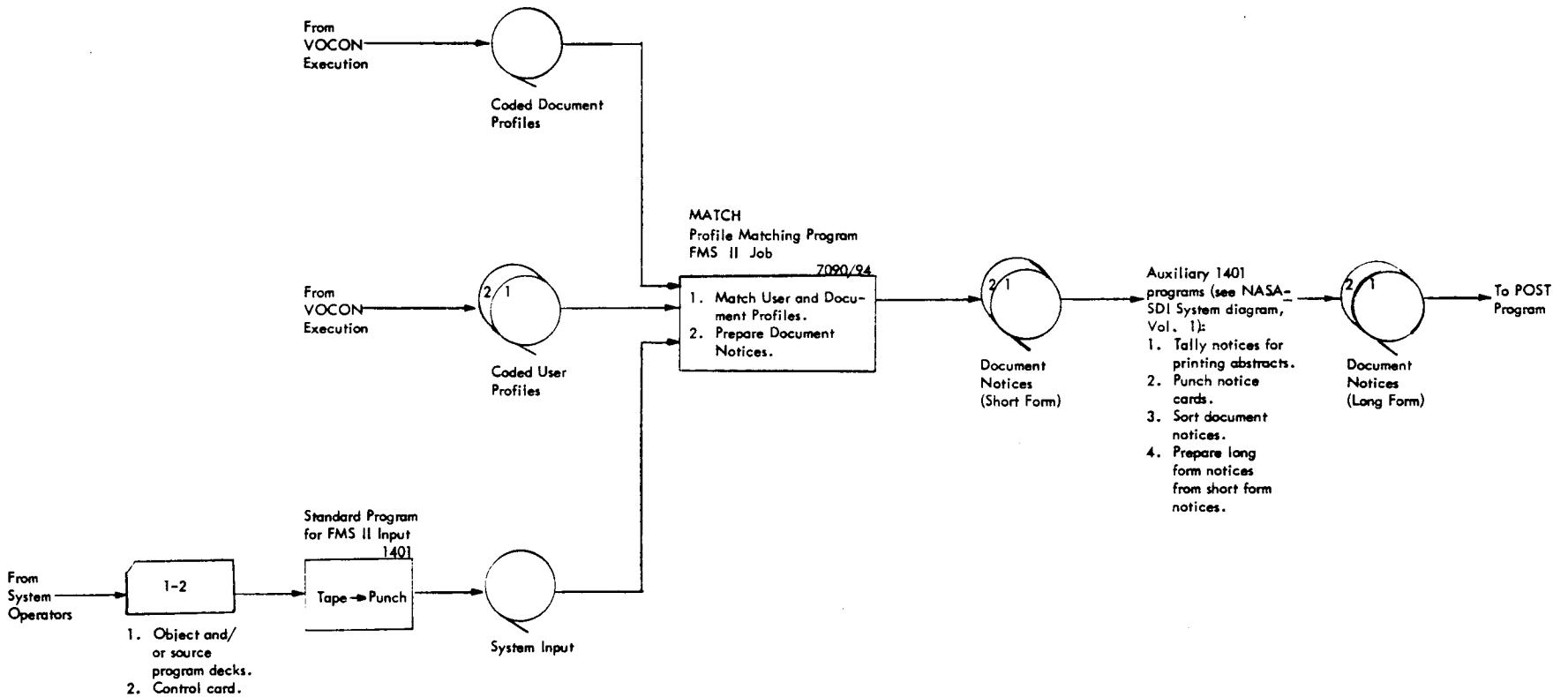


Figure 2. MATCH

CR-62021

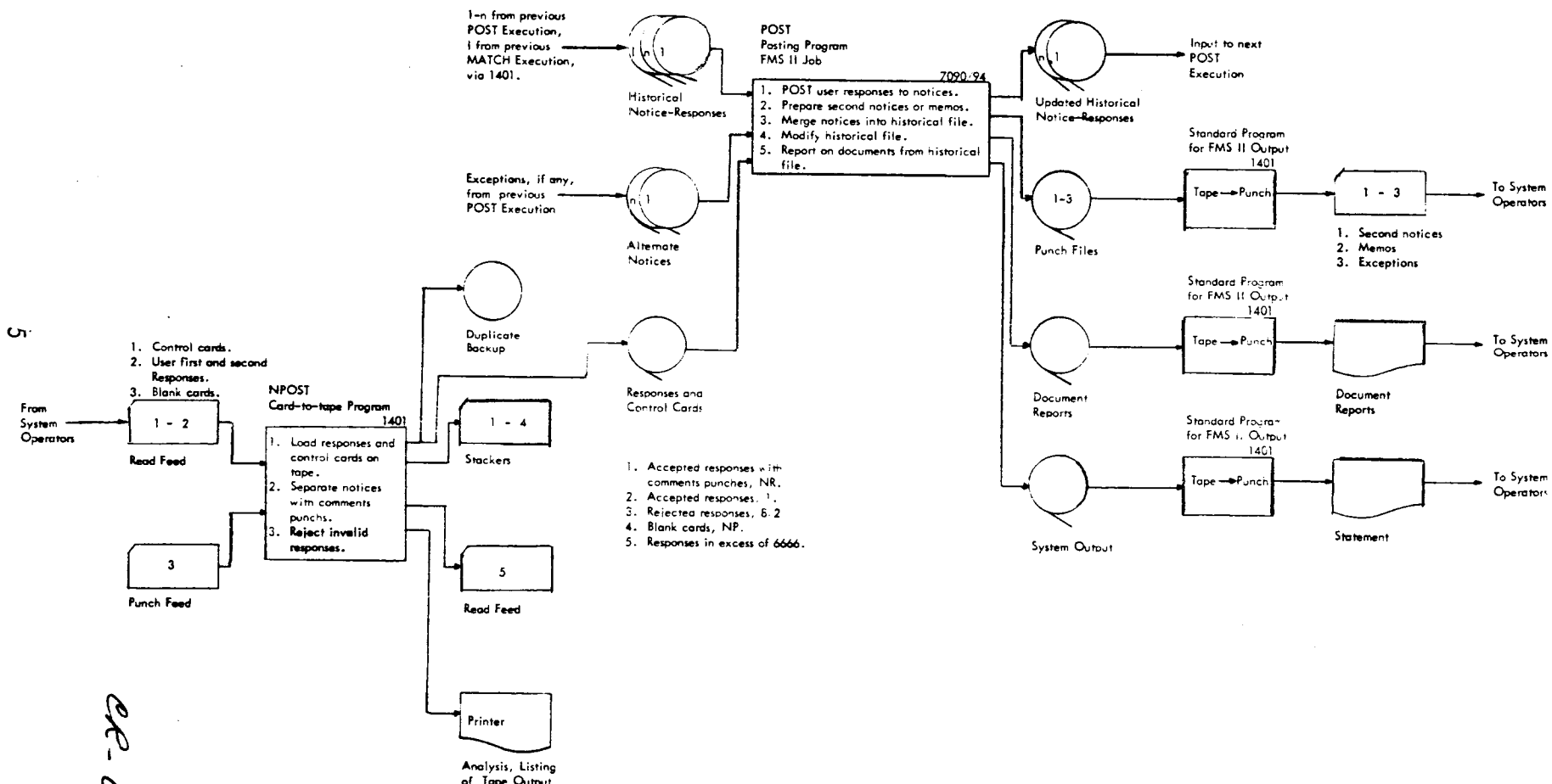


Figure 3. NPOST and POST

CP-62021

B. SDI VOCABULARY CONTROL PROGRAM

1. Program Summary

1.1 Description

The SDI Vocabulary Control Program (VOCON), for the IBM 7090/94 Data Processing System, maintains the vocabulary control guide and prepares and edits user and document profiles for matching. The program runs under a modified version of the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with eight 729 tape units (in addition to FORTRAN units). VOCON is written in FAP and utilizes the FORTRAN input/output package, IOP. It consists of a primary program and several subroutines.

The overall purpose of VOCON, in terms of the SDI system, is to insure that the user and document profiles utilize the same descriptors to identify the same things, so that system accuracy and effectiveness are improved. In other words, descriptors to be used in matching are either accepted or rejected, depending upon whether or not they appear in the vocabulary; but, the vocabulary itself is also accessible to change, as fields of interest appear, expand and contract. Specifically, VOCON updates or originates the vocabulary control guide, updates or originates the user profiles and edits document and user profiles against the vocabulary control guide.

Profile descriptors entering VOCON consist of single words or phrases; user descriptors may also be modified by 'must' or 'not' (for more information on user descriptors, see MATCH). All descriptors in the vocabulary control guide are single words, condensed or coded into single-machine-word values, and the guide is maintained in coded-descriptor order; the English descriptor is also associated with the code. The guide contains primary descriptors that are admissable words (unmodified), secondary descriptors that are synonyms equated to primary descriptors, and trouble terms that are inadmissable words, such as "system". The vocabulary may be changed at any time via word input. The document and user profiles prepared for matching contain only coded descriptors. Optionally, the user profiles may also be maintained in a historical format that includes the English descriptors. The input user profiles enter on cards; the input document profiles are provided on tape via other NASA operations. Phrases input in document profiles are divided into their component words so that the final document profiles are lists of coded single words. These document words also build the vocabulary; any new terms occurring in the documents are automatically added to the vocabulary. Phrases input in the user profiles are maintained as are 'must' and 'not' modifiers, and the component words of phrases are also added to the user profile as single words. User words are not automatically added to the vocabulary.

The processing accomplished by VOCON is divided into nine phases, under the direction of the primary program, MAIN. One or more phases may be dropped from a given execution, depending upon the input to the execution. The purposes of the phases are:

1. Read and process all control cards. Form IO tape usage table. Any error here will cause an error exit with appropriate comment.
2. Read all new input, coding descriptors and forming records on tape A for sorting: vocabulary changes, document profiles, user profile changes, new user profiles. Controlling subroutines GDNRY, GTDOC, GTUSE, CODER.
3. Sort tape A on coded value of descriptors, maintaining profile number (zero if vocabulary change), and item serial number. That is, form an inverted file from tape A. Controlling subroutine SORT.
4. Pass tape A against the vocabulary, simultaneously updating the vocabulary and writing out accepted items, document descriptor items onto tape B and user descriptor items onto tape C. Any vocabulary EQUATE changes will generate modifications for user profiles on tape C. Controlling subroutine UPDATE.
5. Sort tape B on profile number and item serial number. Controlling subroutine SORT.
6. Sort tape C on profile number and item serial number. Controlling subroutine SORT.
7. Create the document profile tape for matching. Controlling subroutine PDTB.
8. Update the user profiles and form the user profile tape for matching. Controlling subroutine UPDUS.
9. Summarize program activity, then exit.

The control cards examined in phase 1 introduce all tape unit assignments and tape number assignments where required, as well as variable program parameters. The presence or absence of control cards and parameters governs the course of the program with respect to creation or updating of various files, etc., or restart. The variable program parameters are:

1.	DATE	Date of computer run
2.	NPHASE	Starting phase for recovery run
3.	MINU	Minimum percentage value of accepted descriptors for retention of user profile
4.	MIND	Minimum percentage value of accepted descriptors for retention of document profile.

1.2 Input

1. Vocabulary control guide
2. Document profiles

3. Coded user profiles
4. Historical user profiles (optional)
5. Control cards, vocabulary changes, new user profiles, user profile changes, user report requests (optional with Item 4).

1.3 Operating Instructions

The SDI VOCON program is a standard FMS system job with a special system tape. The card deck for the system input tape includes in this order: ** job, *ID, * IOP, * XEQ, source decks if any, binary decks if any, * DATA, control cards, vocabulary changes if any, new user profiles if any, user profile changes if any, EOF. The following instructions and comments apply to the three types of runs, initial, update, and restart. Online messages trace the execution and provide operator instructions.

A. Initial Run

Run to create the vocabulary and/or user profile tape(s) and/or document profile tape.

1. At the start of the job the following tapes should be mounted on the drives specified in control card 6:

   a. System input tape
   b. IP document tape (if any)
   c. OP vocabulary tape
   d. OP historical profile tape (if any)
   e. OP user profile tape, reel 1 (if any)
   f. System output tape
   g. For any logical tape number found on card 6, mount a scratch tape on the appropriate drive if none of the tapes mentioned above (a to f) is mounted on it.

2. At the end of phase 4 the OP vocabulary tape will be rewound and unloaded. A message will be printed to mount the second reel of the OP user profile tape or a pool tape if no second reel. The computer will halt until this tape is mounted and START is pressed.

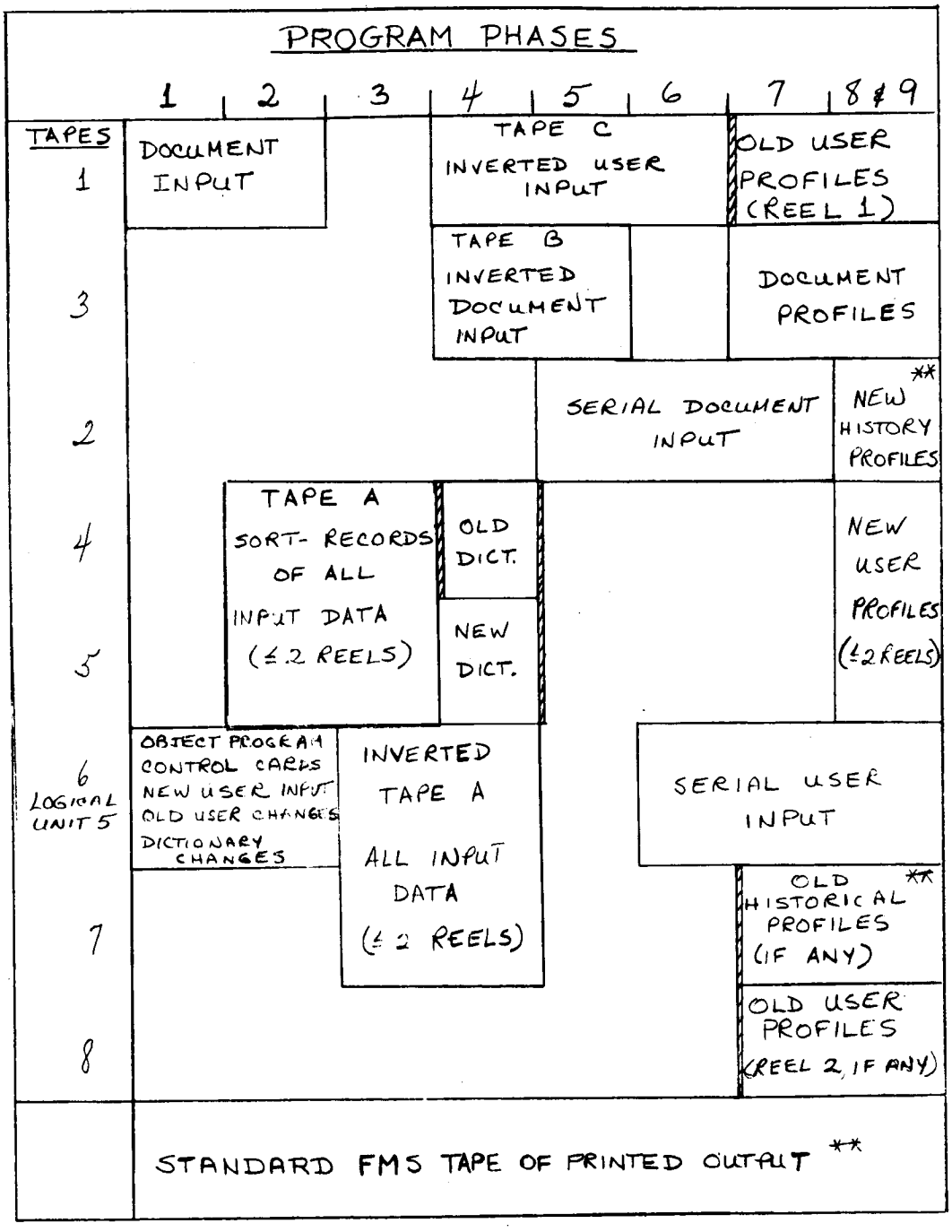
3. At the beginning of phase 7, a message will be printed to mount the IP user profile tape and the first reel of the IP historical user profile tape if these tapes are needed for this run. However, the computer will not stop for this mounting operation.

4. During phase 8, a message to mount the second reel of the IP historical user profile tape will be printed if necessary.

5. During any phase of the program a message to mount a new system output tape will be printed if necessary.

B. Update Run

Run to update the vocabulary tape and/or to create or update the



ODD TAPES → CHANNEL B EVEN TAPES → CHANNEL A IF FORTRAN LOGICAL 5 IS ON CHANNEL A, OTHERWISE WE HAVE THE REVERSE

**LOAD ADDITIONAL TAPES AS REQUIRED

CR-62021

Figure 4. VOCON Tape Usage Scheme

user profile tape and/or to create the document profile tape. The only differences from an initial run are:

1. Instead of reel 1 of the OP user profile tape, a pool tape should be mounted on the specified drive.

2. A message will be printed during phase 3 to mount the IP vocabulary tape on its appropriate drive.

3. At the end of phase 4, both the IP and the OP vocabulary tape will be rewound and unloaded. A message will be printed to mount the two reels of the OP user profile tape. If there are not two reels of OP user profiles the message will indicate that a pool tape is to be mounted.

C. Restart Run

If either an initial run or an update run is terminated before completion of phase 9, the following procedures should be observed:

1. Save all online messages.

2. Remove all tapes noting their respective drives.

3. If the last phase printed online is either phase 1 or phase 2, the job must be rerun from the beginning. A job will be terminated in either of these phases with an online message if a control-card error or a redundant tape exists.

In the first case, the input deck setup should be checked and the missing card inserted or an erroneous card corrected as the error warrants, and the job should then be rerun. In the second case, if the input tape is redundant, a new input tape will have to be generated and the job rerun. If tape A (the tape being written in phase 2) is redundant, a new pool tape should be substituted for that tape, all tapes should be remounted on their respective drives, and the job rerun with the original input tape.

4. If the last phase printed online is phase 3 or beyond:

   a. The IP document profile tape need not be mounted but a pool tape must be mounted in its place.

   b. Vocabulary and/or user cards should be removed from the input deck setup; thus, the last two cards in the deck setup will be control card 8 and the EOF card. It is very important that card 8 contain the additional punches described if there originally was any document and/or user input or vocabulary EQUATE changes. Also, the phase number on card 8 will be the last

phase number appearing in the online messages. These changes, therefore, necessitate the use of a new input tape when restarting any job beyond phase 2.

c. If the job was terminated because an incorrect tape was mounted, a message will be printed indicating this situation. The correct tape should be substituted for this tape and all tapes should then be remounted on their respective drives and the job restarted.

d. If the job was terminated because of a redundant tape during any of the sorting phases (phase 3, phase 5 and phase 6), a new pool tape should be substituted for that tape, all tapes should be remounted on their respective drives and the job restarted using the new input tape.

5. In phases 4, 7 and 8 where a restart job needs a new input tape and also requires the tape generated on the FMS system input tape unit, change control card 6 in cols. 37-38 to the logical tape number for a drive not being used, mount the generated tape on this drive and mount the new input tape on the FMS system input tape unit.

1.4 Output

1. Vocabulary control guide, updated

2. Coded document profiles

3. Coded user profiles, updated

4. Historical user profiles, updated (optional)

5. Printout of vocabulary acceptances and rejections, user and document profile acceptances and rejections, user reports (optional with Item 4).

2. Record Formats

2.1 Input

1. Vocabulary Control Guide

The vocabulary control guide consists of 101 files. File 1 is a label. Files 2-101 are vocabulary records sorted on coded descriptor value.

a. File 1

CR-62021

Record	
1	label of 12 words (72 characters)
2	catalog of entries: 100 consecutive pairs of computer words corresponding to the upper and lower limits of a corresponding data file. The first entry in this catalog will always be a word of all zeros. For files having no entries, both limits will be computer words of all sevens.
3	100 computer words each of which contains the number of logical records in the corresponding file.

b. Files 2, 3, 4, ..., 101

Each of these files contains approximately 1/100th of the entire vocabulary. Each physical record is 450 words long consisting of 15 descriptor records each 30 words in length. Each descriptor record has the following format:

Word	
1	coded descriptor
2	coded descriptor. If word 1 = word 2, this is a primary descriptor. If word 2 is zero, this is a trouble term. Word 1 may never equal zero.
3	date of first appearance in any document profile or zero.
4	date of first appearance in any user profile or zero.
5	number of documents having this descriptor or zero.
6	number of users having this descriptor or zero.
7-26	alphanumeric descriptor, not over 120 characters or blanks.
27-30	not used. All zeros.

2. Document Profiles

This file is provided via other NASA operations, from which its format can be obtained. It is sorted on document number.

3. Coded User Profiles

The coded user profile tape consists of three files. Files 1 and 3 are labels. File 2 is sorted on user number and coded descriptor.

a. File 1 — One record

Char.	
1-65	tape identification
66	tape number (1, 2, ...)
67-72	date tape was created

b. File 2 — One record per user $\leq$ 5000 words

Word	
1-24	user header, card image
25	date of entry to system
26	date of last change to record
27	Bit
	1-12 number of single-word coded descriptors in sorted order, N
	13-24 number of added on latest run, M, not sorted
	25-36 number of deleted single-word descriptors, L
28-30	not used
31	reserved to indicate presence of 'not' descriptors

Following are M + N words of single-word coded descriptors:

Bit	
S-2	Octal 0, 'may' descriptor Octal 1, 'must' descriptor Octal 2, 'must' author Octal 3, 'must' contract number Octal 4, 'not' subject field Octal 5, 'not' author Octal 6, 'not' contract number Octal 7, 'not' descriptor
3-32	These thirty bits contain the coded value of the descriptor word or phrase.
33-35	For single-word descriptors, an octal 0 indicates deleted word, 1 that descriptor is used only as a single word, 2 that descriptor is used only in a phrase, 3 that descriptor is used both as a single word and in a phrase, 4 that descriptor is a special descriptor.

The remainder of the record consists of the coded values of the phrase descriptors and pointers to corresponding word locations in the single-word area. A given phrase would consist of one word of code, followed by as many words as are needed to contain the pointers, three to a word.

c. File 3 One record

Word	
1	Number of next user tape (0 if zero is last or only tape, otherwise 1, 2, ..)
2-14	Not used

4. Historical User Profiles (Optional)

The historical user profiles consist of three files. Files 1 and 3 are

CP-62021

labels. File 2 is sorted on user number.

a. File 1 One record

Char.	
1-65	Tape identification
66	Tape number (1, 2, ...)
67-72	Data tape was created

b. File 2 One record per user $\leq$ 1000 words

Record Format:

Word	
1-24	User header, card image
25	Date of entry to system
26	Date of deletion from system (zero if active)

Following is one logical record per descriptor in this format:

Word	
1	Coded value of descriptor
2	Date descriptor was added to profile
3	Date of deletion of descriptor (zero if active)
4	Length of this logical record, N
5 thru N-3	BCD descriptor, as many words as needed

c. File 3 One record

Word	
1	Number of next historical profiles tape (zero if last or only tape; otherwise 2, 3, ...)
2-14	Not used

5(a). Control Cards

Card 1	
Cols. 1-6	Label of input vocabulary. Slashes in 1-6 if no such input tape
Card 2	
Cols. 1-72	Label of output vocabulary control guide (used only if no input vocabulary file.)
Card 3	
Cols. 1-72	Label of output coded user profiles
Card 4	
Cols. 1-72	Label of input coded user profiles. Slashes in 1-6 if no such input tape.
Card 5	
Cols. 1-72	Label of input and output historical user profiles. Card 5 is blank if no historical profiles.
Card 6	Input tape unit assignment. First four tapes on one channel, next four on second channel.

A blank or zero assignment is not permitted.

Cols. 1-2	Input document profiles and input coded user profiles, reel 1
7-8	Output document profiles
13-14	Output vocabulary and output coded user profiles, reel 2
19-20	Input historical user profiles
25-26	Output historical user profiles
31-32	Input vocabulary and output coded user profiles, reel 1
37-38	System input tape
43-44	Input coded user profiles, reel 2
49-50	System output tape
Card 7	Tape numbers for which mounting will be requested on line. A blank indicates that no tape exists for this file. Tapes not mentioned are mounted at start of run.
Cols. 8-12	Input vocabulary
14-18	Input coded user profiles, reel 1
20-24	Input coded user profiles, reel 2
26-30	Input historical user profiles, reel 1
32-36	Input historical user profiles, reel 2
38-42	Output coded user profiles, reel 1
44-48	Output coded user profiles, reel 2
Card 8	Program parameters
Cols. 1-6	Date; numeric month, day, year
17-18	Phase number at which restart is to begin
22-24	Minimum percentage value of accepted descriptors for retention of user profile
28-30	Minimum percentage value of accepted descriptors for retention of document profile

If job is restarted, add to card 8 the following:

Col. 14	1 if document profile input, blank if none
16	1 if coded user profile input or vocabulary EQUATE changes, blank if neither

Following the eight control cards are cards for vocabulary and user changes.

Following those, if any, is one of two cards:

Cols. 1-6	NODOCT if there is no input document profile tape.
1-7	DOCTAP1 if there is an input document profile tape. DOCTAP2 if there are two input document profile tapes.

5(b). Vocabulary Changes

Card 1	
Cols. 1-4	DICT

CR-62021

Descriptors may be added to or deleted from the vocabulary, classified as trouble terms, or equated as synonyms. The trouble terms and equate changes will also add descriptors to the vocabulary if the descriptors involved are not already there.

a. Add Changes

Card 1		
Cols.	1-60	Alphameric descriptor
	71	A
	72	Blank if no second card, C if second card follows
Card 2		
Cols.	1-60	Alphameric descriptor continued
	71	A
	72	Blank

b. Delete Changes

Same as add changes, except D in col. 71.

c. Trouble Term Change

Same as add changes, except * in col. 71.

d. Equate Change (will cause updating of user profiles)

Card	1	Secondary
Cols.	1-60	Alphameric descriptor, secondary
	71	E
	72	Blank if no second card, C if second card follows
Card 2		Secondary
Cols.	1-60	Alphameric descriptor, secondary continued
	71	E
	72	Blank
Card 3		Primary
Cols.	1-60	Alphameric descriptor, primary
	71	T
	72	Blank if no second card, C if second card follows
Card 4		Primary
Cols.	1-60	Alphameric descriptor, primary, continued
	71	T
	72	Blank
Last Card		
Cols.	1-6	ENDICT

5(c). New user profiles and user profile changes.

Card 1
Cols. 1-4 USER

a. New Users - Header

Card 1

Cols.		
	1- 6	Profile number
	8, 10	User's initials
	12-30	User's surname
	36-38	Location
	44-48	Employee number (optional)
	52	Corporate security clearance
	53	Federal security clearance
	70	U
	71	H
	72	Blank if one header card, otherwise C

Card 2

Cols.		
	1- 6	Same as first card
	7-66	Address for outside mailing (left adjusted)
	70-71	Same as first card
	72	Blank

b. New Users - Descriptors

Card 1

Cols.		
	1-60	Alphameric descriptor
	61-66	Profile number
	70	Descriptor usage code
	71	Blank
	72	C if second card follows, blank if no second card

Card 2

Cols.		
	1-60	Alphameric descriptor, continued
	61-66	Profile number
	70	Descriptor usage code
	71	Blank
	72	Blank

Where descriptor usage code is:

M	'must' ordinary descriptor
N	'not' ordinary descriptor
C	'must' contract-number
D	'not' contract number
P	'must' author
Q	'not' author
S	'not' subject field

c. Add Changes - Header
Same as new users

d. Add Changes - Descriptors
Same as new users, except A in col. 71.

e. Delete Changes - Descriptors
Same as new users, except D in col. 71.

CR-62021

f. Delete entire user profile

Cols.	1-60	Blank
	61-66	Profile number
	67-70	Blank
	71	D
	72	U

Last Card

Cols.	1-6	ENDUSR

If historical user profiles are kept, user reports may be called for between the USER and ENDUSR cards.

g. User Reports

Cols.	1- 6	Lower profile limit of report, profile number
	8-13	Upper profile limit of report, profile number, or blank if only one profile (col. 1-6) to be reported
	71	R
	72	1 if only active descriptors are to be reported 2 if all descriptors are to be reported

2.2 Output

1. Vocabulary control guide, updated - same format as input.
2. Coded document profiles - same format as coded user profiles.
3. Coded user profiles, updated - same format as input.
4. Historical user profiles, updated (optional) - same format as input.
5. Operational comments - see example following program listing.

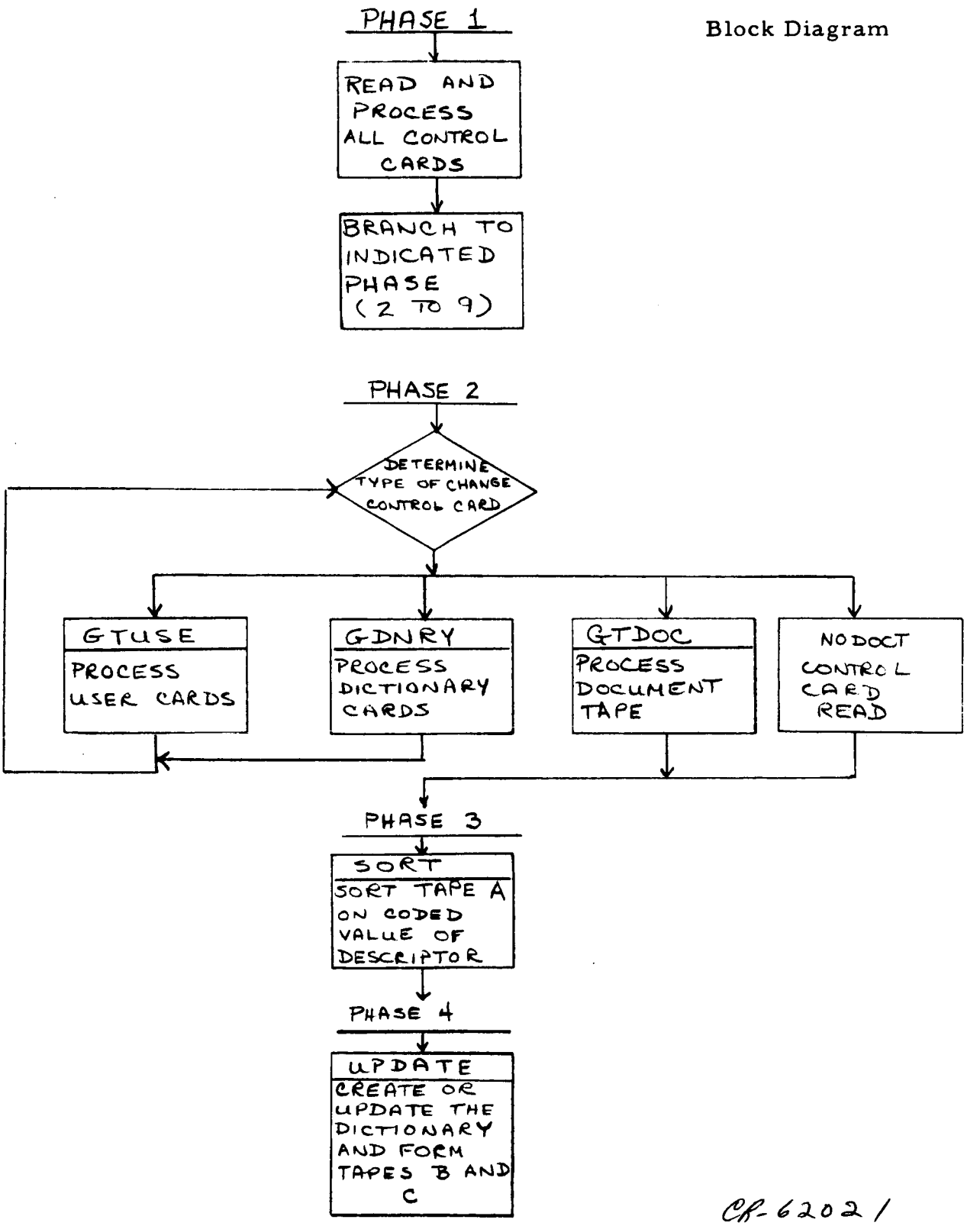
CR-62021

MAIN PROGRAM

This program is divided into nine phases. In phase 1, all control cards are read and processed. In phase 2, user cards are processed by GTUSE, vocabulary cards by GDNRY, and the document tape by GTDOC. The output tape resulting from this phase is called tape A. During phase 3, tape A is sorted on the coded value of the descriptors. Phase 4 uses the UPDATE subroutine to create or update the vocabulary tape while at the same time forming document tape (tape B) and/or a user tape (tape C). Phase 5 sorts the document tape according to profile and serial number, and phase 6 sorts the user tape according to profile and serial number. During phase 7, a document profile tape is created. During phase 8, a user profile tape is created or updated, and also an historical user profile tape may be created or updated. Phase 9 summarizes program activity, then exits. The subroutine of the program follow, in alphabetical order.

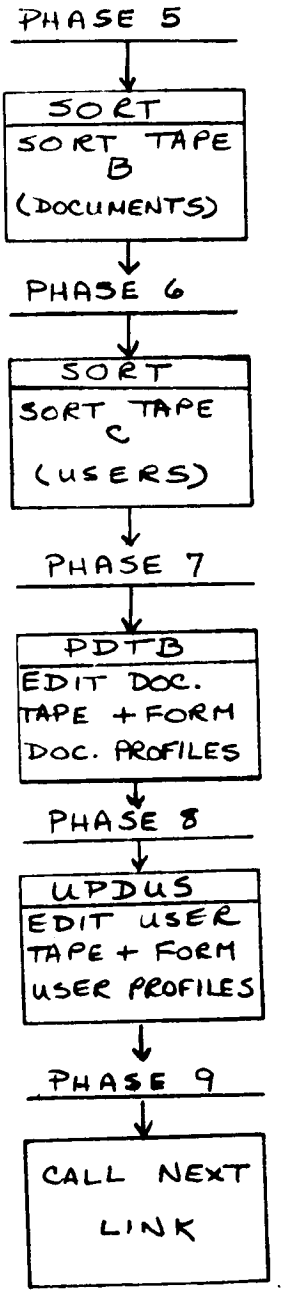
CR-62021

Block Diagram

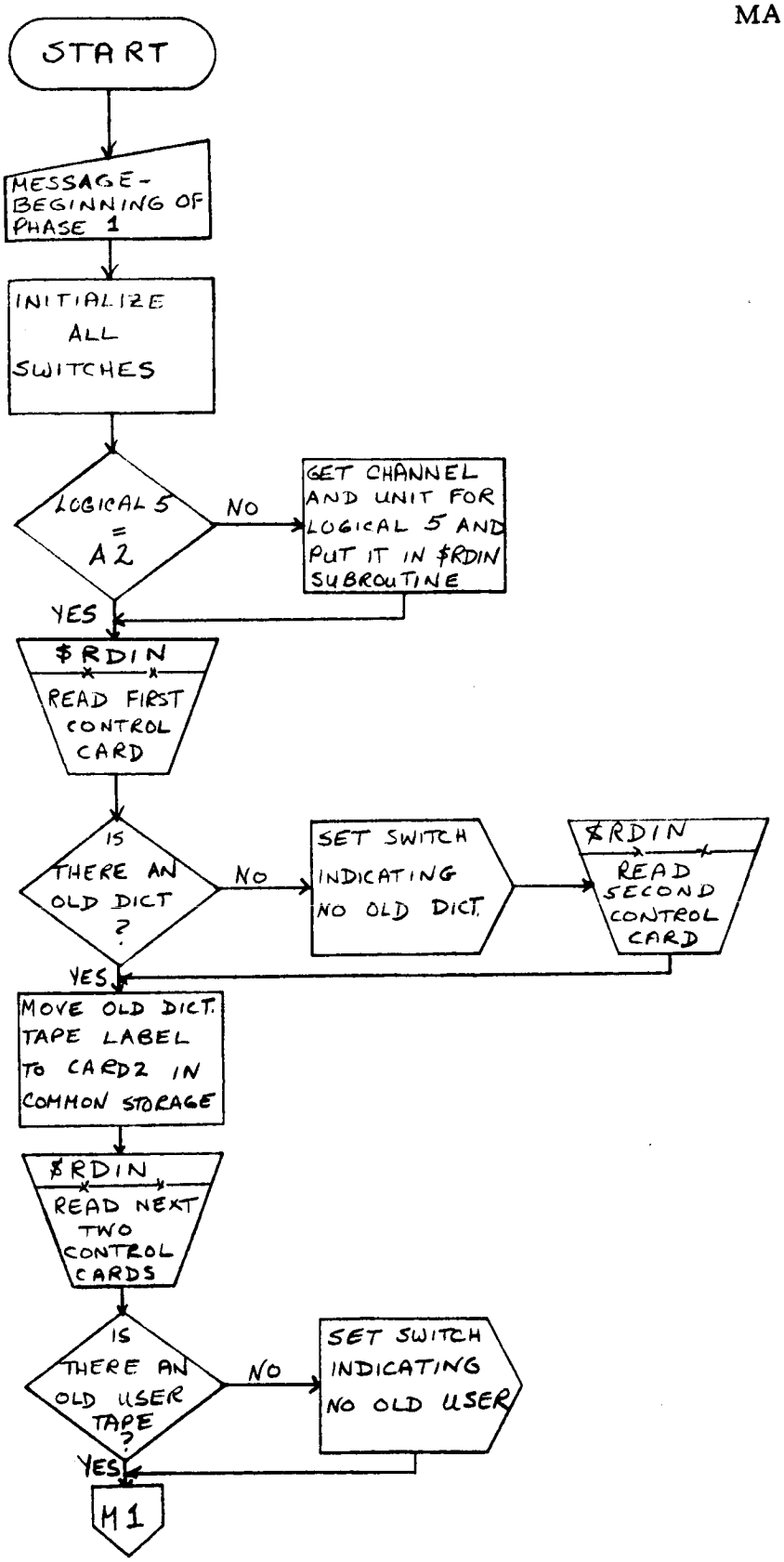


CR-6202/

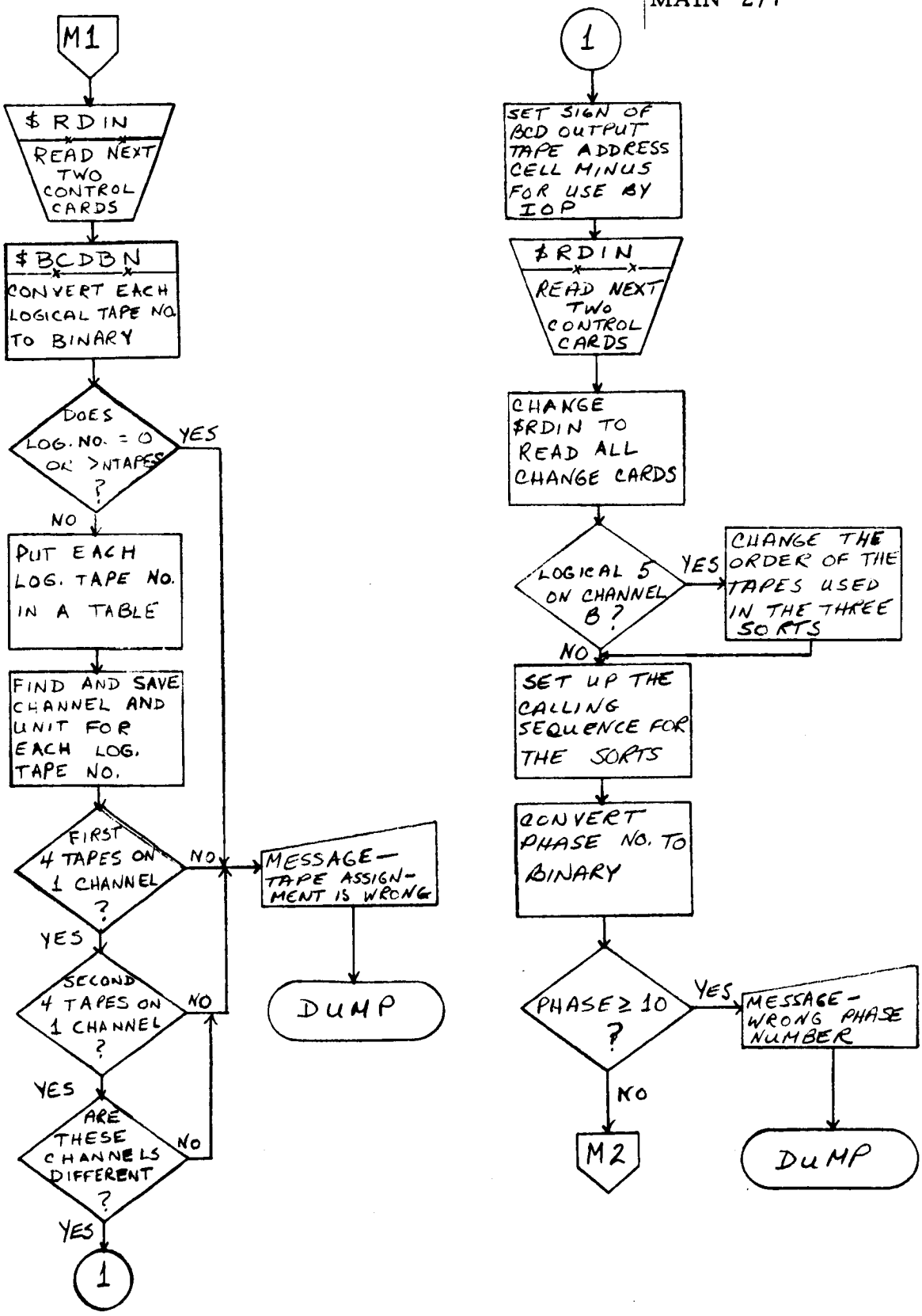
Block Diagram



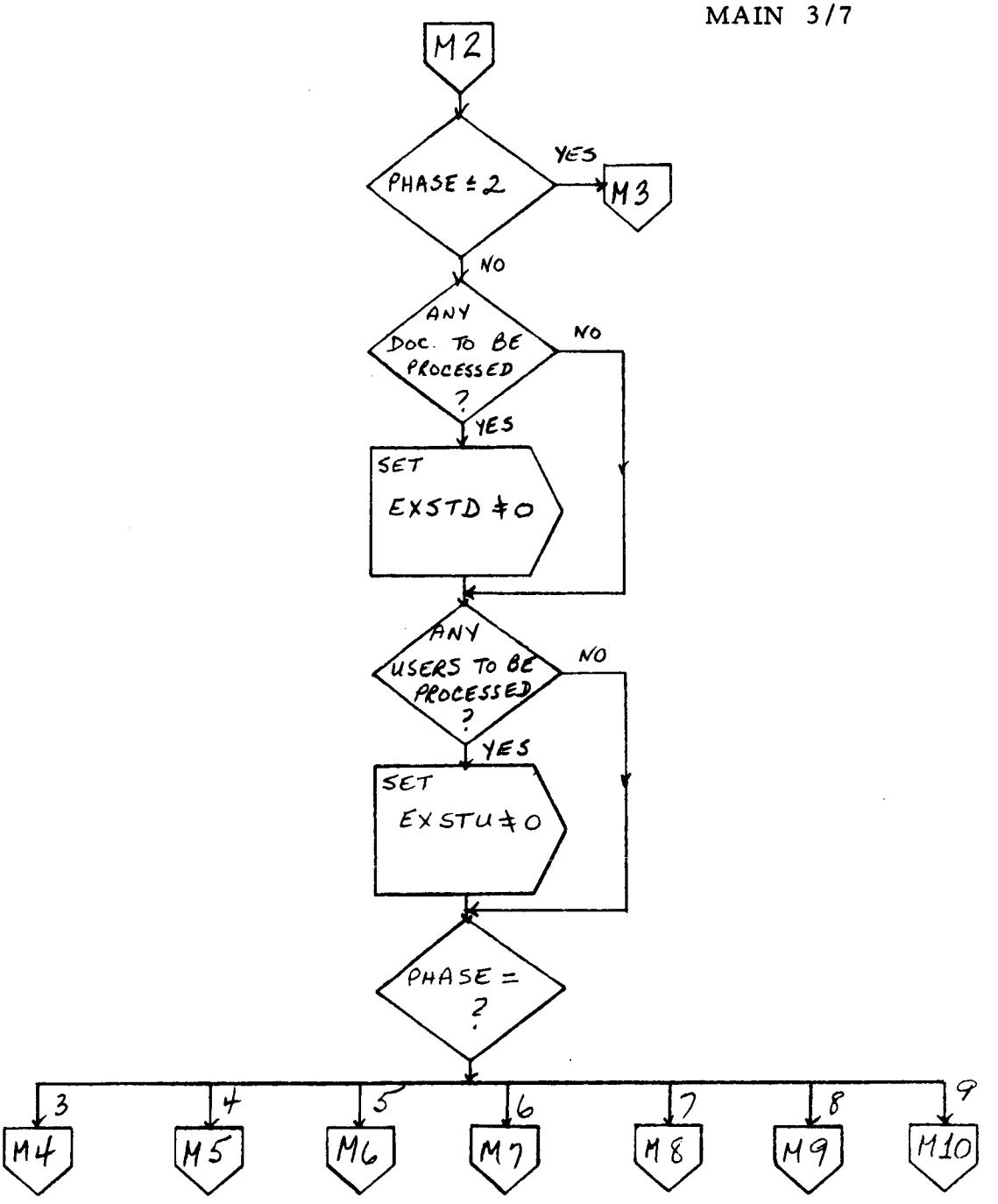
CR-62021


START
MESSAGE - BEGINNING OF PHASE 1
INITIALIZE ALL SWITCHES
LOGICAL 5 = A2
NO
GET CHANNEL AND UNIT FOR LOGICAL 5 AND PUT IT IN $RDIN SUBROUTINE
YES
$RDIN
READ FIRST CONTROL CARD
IS THERE AN OLD DICT ?
NO
SET SWITCH INDICATING NO OLD DICT.
$RDIN
READ SECOND CONTROL CARD
YES
MOVE OLD DICT. TAPE LABEL TO CARD2 IN COMMON STORAGE
$RDIN
READ NEXT TWO CONTROL CARDS
IS THERE AN OLD USER TAPE ?
NO
SET SWITCH INDICATING NO OLD USER
YES
M1

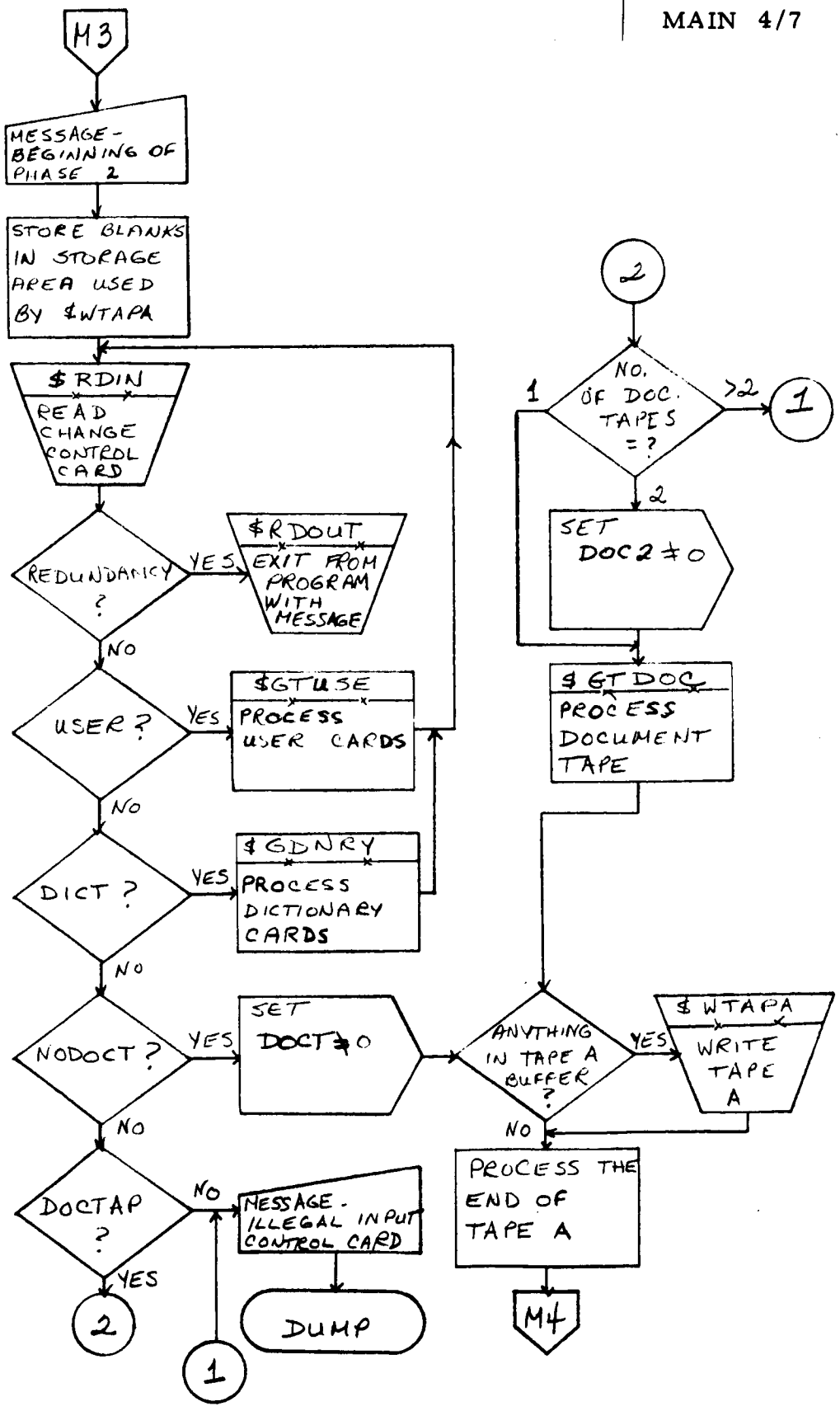
CR-6202/


M1
$RDIN
READ NEXT TWO CONTROL CARDS
$BCDBN
CONVERT EACH LOGICAL TAPE NO. TO BINARY
DOES LOG. NO. = 0 OR >NTAPES ?
YES
NO
PUT EACH LOG. TAPE NO. IN A TABLE
FIND AND SAVE CHANNEL AND UNIT FOR EACH LOG. TAPE NO.
FIRST 4 TAPES ON 1 CHANNEL ?
NO
MESSAGE— TAPE ASSIGN-MENT IS WRONG
DUMP
YES
SECOND 4 TAPES ON 1 CHANNEL ?
NO
YES
ARE THESE CHANNELS DIFFERENT ?
NO
YES
1
1
SET SIGN OF BCD OUTPUT TAPE ADDRESS CELL MINUS FOR USE BY IOP
$RDIN
READ NEXT TWO CONTROL CARDS
CHANGE $RDIN TO READ ALL CHANGE CARDS
LOGICAL 5 ON CHANNEL B ?
YES
CHANGE THE ORDER OF THE TAPES USED IN THE THREE SORTS
NO
SET UP THE CALLING SEQUENCE FOR THE SORTS
CONVERT PHASE NO. TO BINARY
PHASE ≥ 10 ?
YES
MESSAGE— WRONG PHASE NUMBER
NO
M2
DUMP

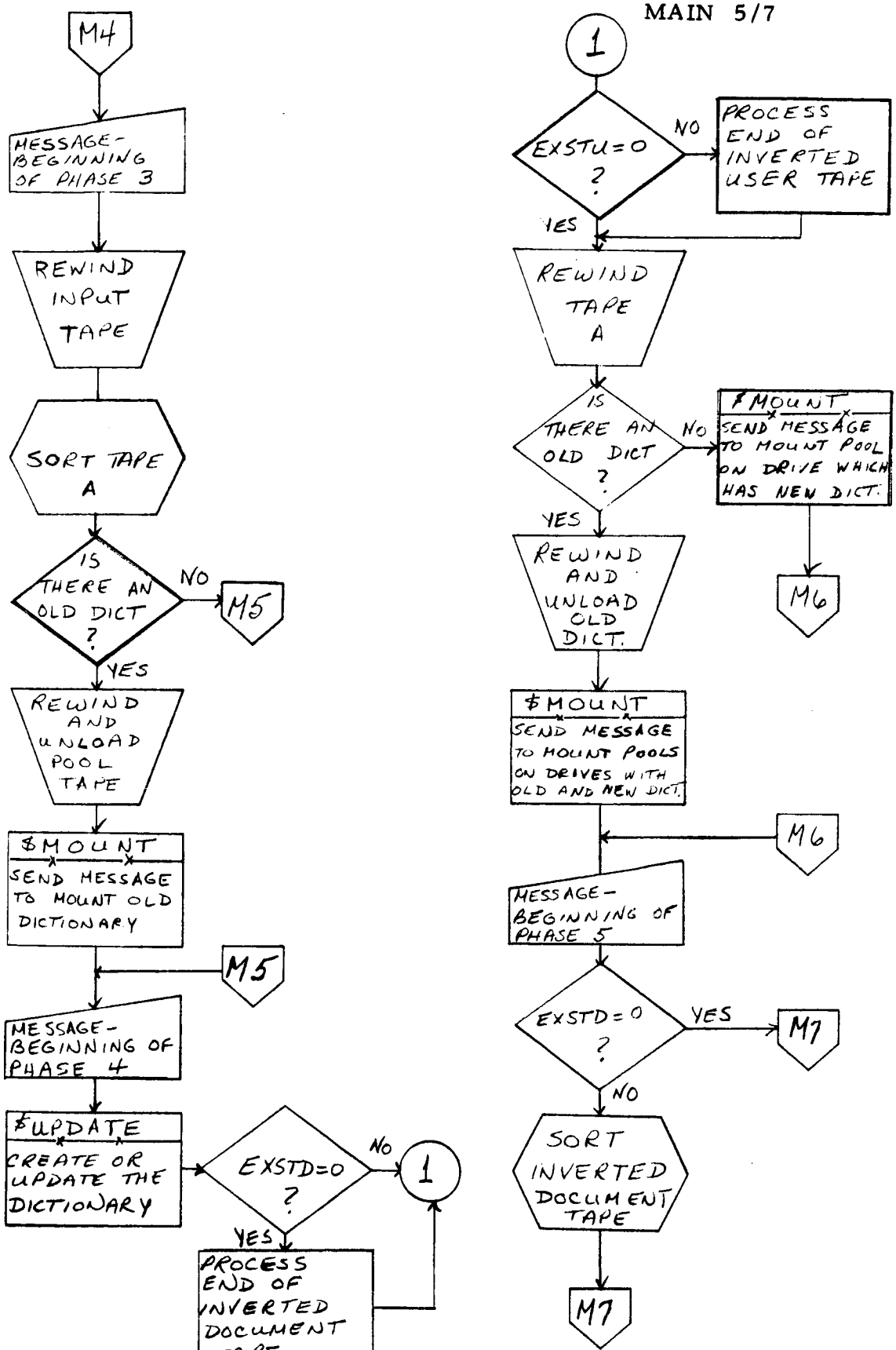
CR. 62021


M2
PHASE ≠ 2
YES
M3
NO
ANY DOC. TO BE PROCESSED ?
NO
YES
SET EXSTD ≠ 0
ANY USERS TO BE PROCESSED ?
NO
YES
SET EXSTU ≠ 0
PHASE = ?
3
4
5
6
7
8
9
M4
M5
M6
M7
M8
M9
M10

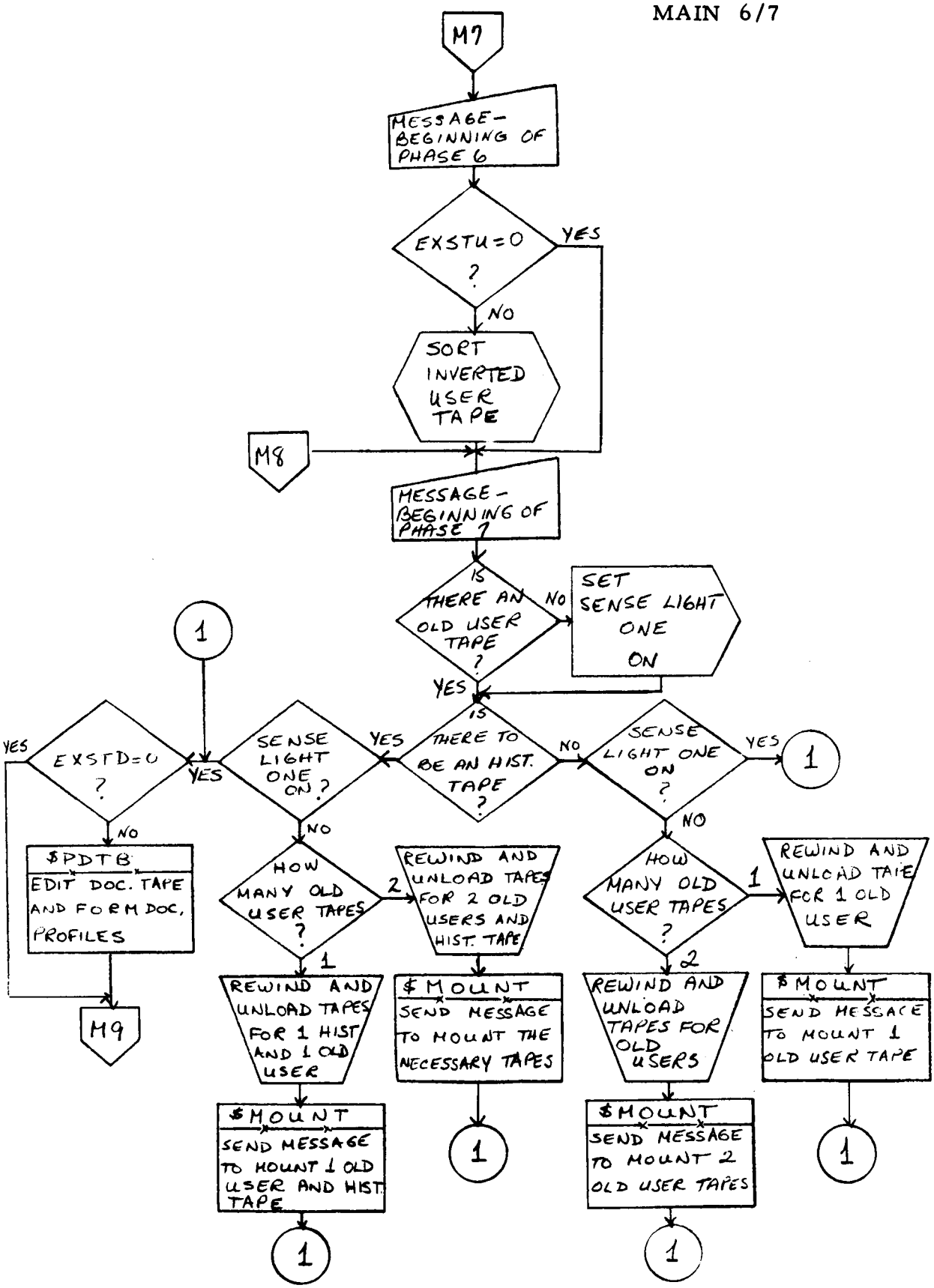
CR-6202/


M3
MESSAGE - BEGINNING OF PHASE 2
STORE BLANKS IN STORAGE AREA USED BY $WTAPA
$RDIN
READ CHANGE CONTROL CARD
REDUNDANCY ?
YES
$RDOUT
EXIT FROM PROGRAM WITH MESSAGE
NO
USER ?
YES
$GTUSE
PROCESS USER CARDS
NO
DICT ?
YES
$GDNRY
PROCESS DICTIONARY CARDS
NO
NODOCT ?
YES
SET DOCT ≠ 0
NO
DOCTAP ?
NO
MESSAGE - ILLEGAL INPUT CONTROL CARD
DUMP
YES
2
1
2
NO. OF DOC. TAPES = ?
1
>2
1
2
SET DOC2 ≠ 0
$GTDOC
PROCESS DOCUMENT TAPE
ANYTHING IN TAPE A BUFFER ?
YES
$WTAPA
WRITE TAPE A
NO
PROCESS THE END OF TAPE A
M4

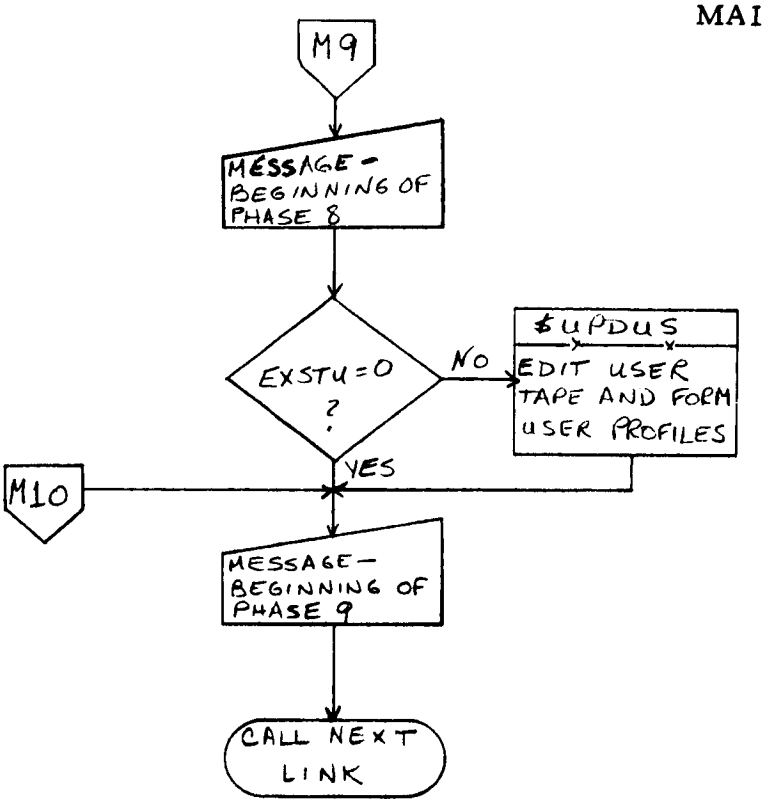
CR-62021


M4
MESSAGE - BEGINNING OF PHASE 3
REWIND INPUT TAPE
SORT TAPE A
IS THERE AN OLD DICT ?
NO
M5
YES
REWIND AND UNLOAD POOL TAPE
$MOUNT
SEND MESSAGE TO MOUNT OLD DICTIONARY
M5
MESSAGE - BEGINNING OF PHASE 4
$UPDATE
CREATE OR UPDATE THE DICTIONARY
EXSTD=0 ?
NO
1
YES
PROCESS END OF INVERTED DOCUMENT TAPE
1
EXSTU=0 ?
NO
PROCESS END OF INVERTED USER TAPE
YES
REWIND TAPE A
IS THERE AN OLD DICT ?
NO
$MOUNT
SEND MESSAGE TO MOUNT POOL ON DRIVE WHICH HAS NEW DICT.
M6
YES
REWIND AND UNLOAD OLD DICT.
$MOUNT
SEND MESSAGE TO MOUNT POOLS ON DRIVES WITH OLD AND NEW DICT.
M6
MESSAGE - BEGINNING OF PHASE 5
EXSTD=0 ?
YES
M7
NO
SORT INVERTED DOCUMENT TAPE
M7

CR-6208/


M7
MESSAGE - BEGINNING OF PHASE 6
EXSTU=0 ?
YES
NO
SORT INVERTED USER TAPE
M8
MESSAGE - BEGINNING OF PHASE 7
IS THERE AN OLD USER TAPE ?
NO
SET SENSE LIGHT ONE ON
YES
1
IS THERE TO BE AN HIST. TAPE ?
YES
NO
SENSE LIGHT ONE ON ?
YES
1
SENSE LIGHT ONE ON.?
YES
NO
EXSTD=0 ?
YES
NO
$PDTB
EDIT DOC. TAPE AND FORM DOC. PROFILES
M9
HOW MANY OLD USER TAPES ?
2
1
REWIND AND UNLOAD TAPES FOR 2 OLD USERS AND HIST. TAPE
$MOUNT
SEND MESSAGE TO MOUNT THE NECESSARY TAPES
1
HOW MANY OLD USER TAPES ?
1
2
REWIND AND UNLOAD TAPE FOR 1 OLD USER
$MOUNT
SEND MESSAGE TO MOUNT 1 OLD USER TAPE
1
REWIND AND UNLOAD TAPES FOR 1 HIST AND 1 OLD USER
$MOUNT
SEND MESSAGE TO MOUNT 1 OLD USER AND HIST TAPE
1
REWIND AND UNLOAD TAPES FOR OLD USERS
$MOUNT
SEND MESSAGE TO MOUNT 2 OLD USER TAPES
1

CR-62021

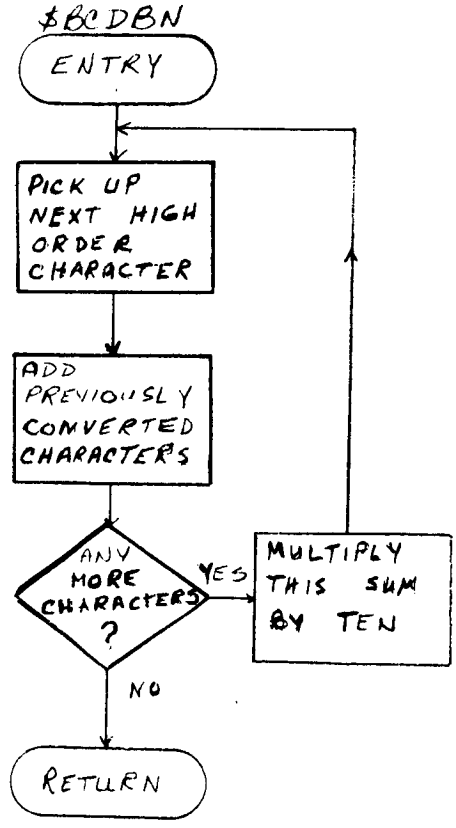

M9
MESSAGE - BEGINNING OF PHASE 8
EXSTU=0 ?
NO
$UPDUS
EDIT USER TAPE AND FORM USER PROFILES
YES
M10
MESSAGE - BEGINNING OF PHASE 9
CALL NEXT LINK

CP-62021

Subroutine BCDBN

This subroutine converts a BCD integer into a binary number.

CR-62021


$BCDBN
ENTRY
PICK UP NEXT HIGH ORDER CHARACTER
ADD PREVIOUSLY CONVERTED CHARACTERS
ANY MORE CHARACTERS ?
YES
MULTIPLY THIS SUM BY TEN
NO
RETURN

QR_62021

Subroutine CODER

The calling sequence of CODER is

```
        TSX     $CODER,4
        TSX     TYPE,0
        TSX     BUFIN,0
        TSX     BUFOUT,0
        TSX     K,0
```

where

TYPE indicates source of input
- 0 vocabulary change
- 1 document descriptor
- 3 new user descriptor
- 4 user change descriptor.

BUFIN is location of start of input buffer.

BUFOUT is location of start of output buffer. BUFIN and BUFOUT arrays are in forward, FAP order.

K is the number of sort items placed in BUFOUT by CODER. If the input descriptor consists of n alphameric words, then K = n if n = 1, or K = n + 1 if n > 1.

CODER expects 72 characters per input card image, the alphameric cols. 1-60. Profile header cards are not input to CODER. Each descriptor input to CODER will generate K items of the format:

Word

1 coded value of input descriptor

2 alphameric profile number (zero if TYPE = 0)

3 miscellaneous:

Bit	
S- 2	zero
3- 5	same as TYPE
6-11	same as col. 71 of input card
12-13	0 do not add to vocabulary 1 add to vocabulary
14-15	0 do not add to profile if code not on vocabulary 1 add code to profile 3 add code to profile descriptor list
16	reserved for use by subroutine UPDATE
17	determined by the subroutine that calls CODER. 1 if code for word within phrase 0 if single word descriptor or coded phrase
18-35	zero.

4 contains L, length of item in computer words.

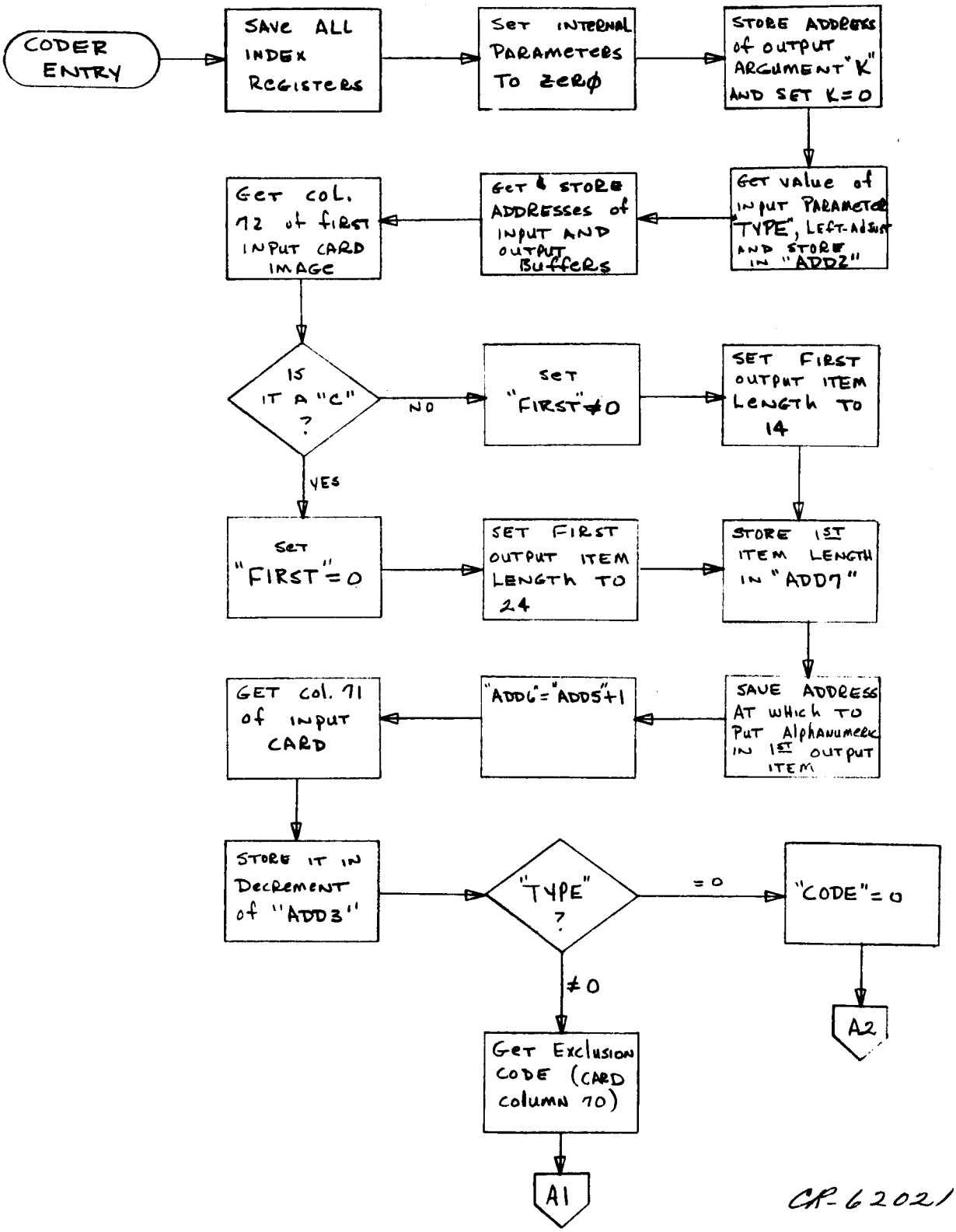
5-L alphameric version of the coded input.

CR-62021

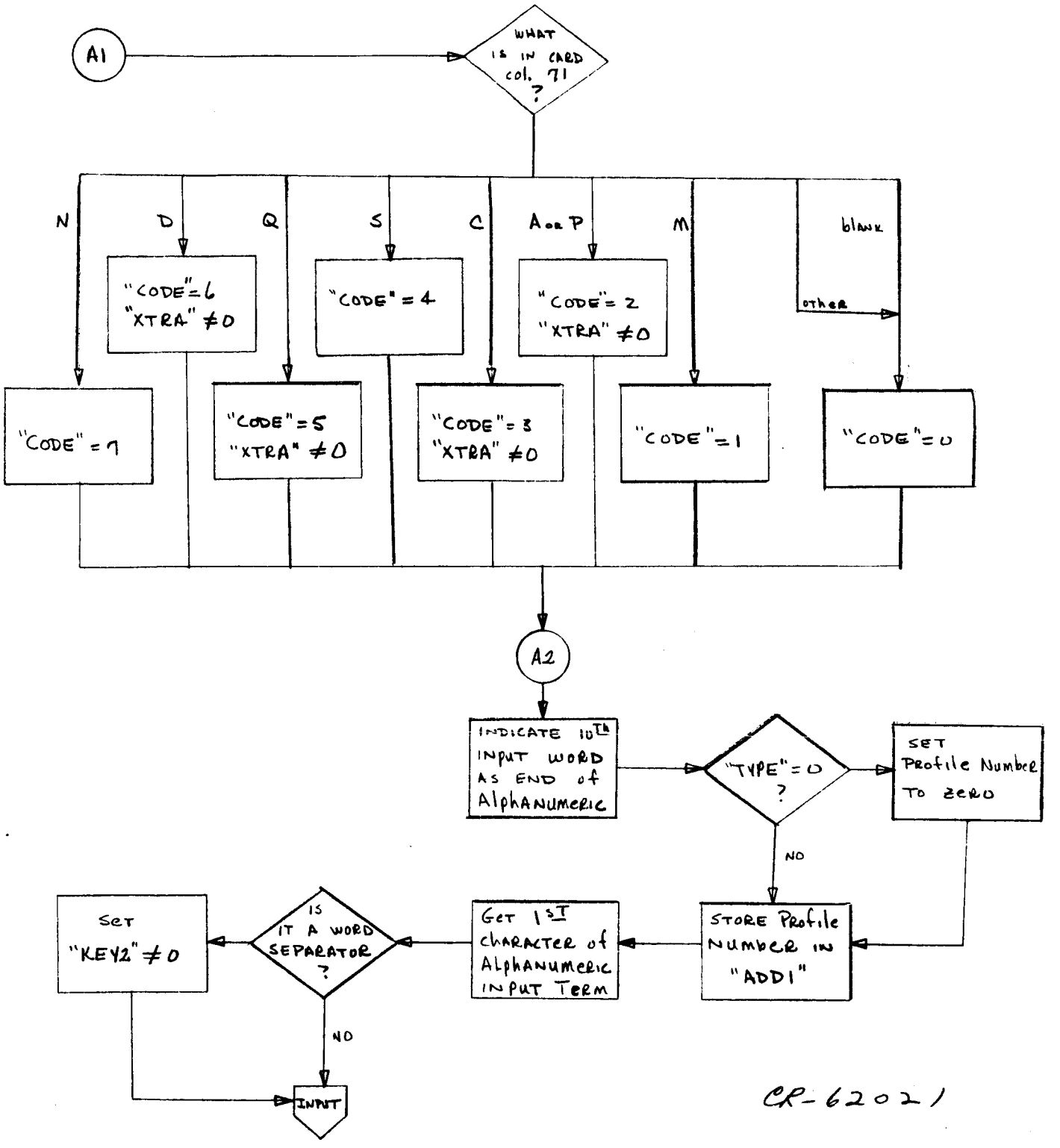
If K>1, the second and following output items are four (L = 4) words in length and have no alphameric version. The first item contains the alphameric version and is 14 or 24 words in length (L = 14 or 24). If a descriptor is illegal, i.e., contains an illegal character, then K = 1, word 1 of the item is all zero, and word 3 is set:

Bit	
12-14	zero
18-35	77777)8

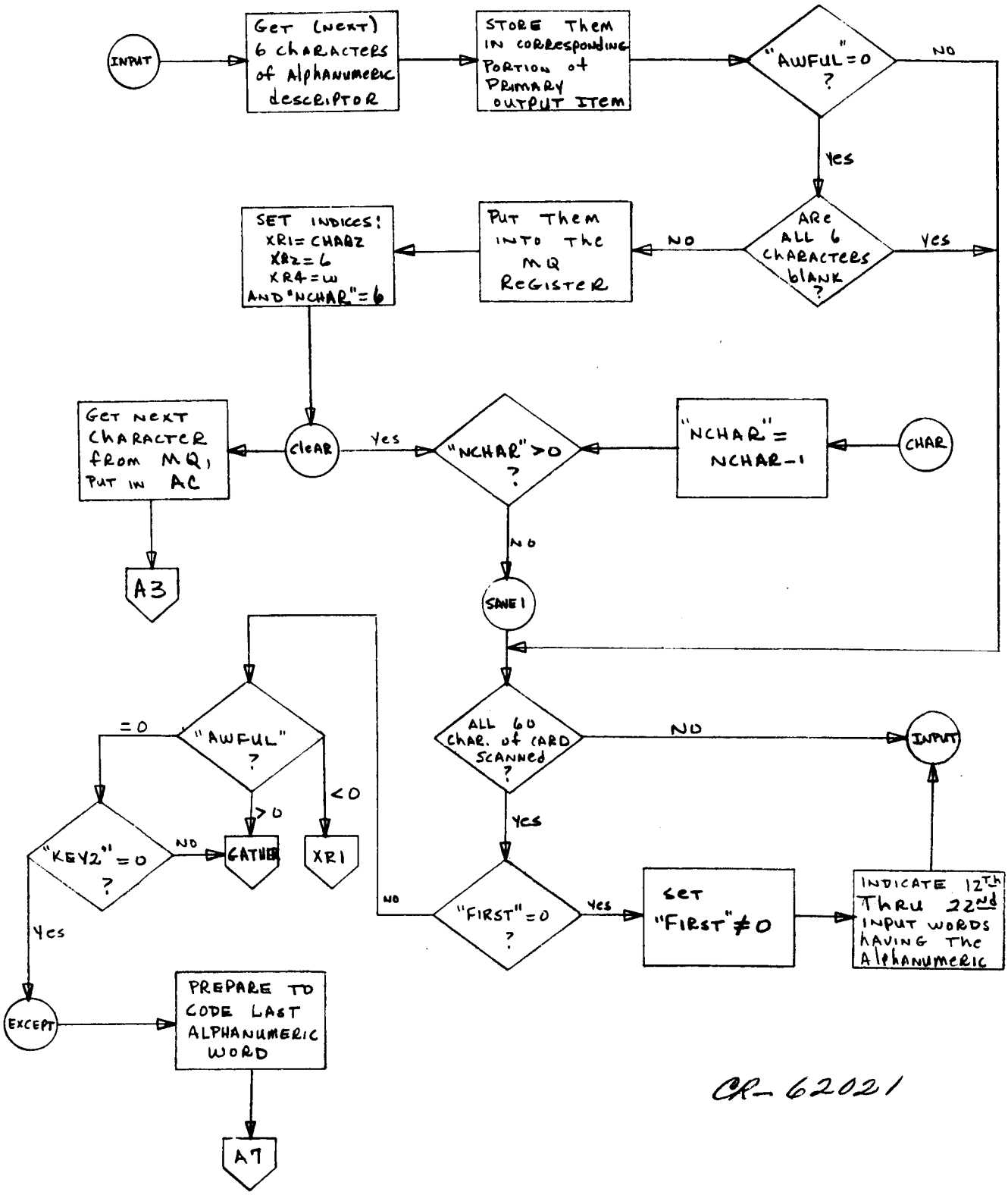
CR-62021


CODER ENTRY
SAVE ALL INDEX REGISTERS
SET INTERNAL PARAMETERS TO ZERO
STORE ADDRESS of OUTPUT ARGUMENT "K" AND SET K=0
GET value of INPUT PARAMETER "TYPE", LEFT-ADJUST AND STORE IN "ADD2"
GET & STORE ADDRESSES of INPUT AND OUTPUT Buffers
GET col. 72 of FIRST INPUT CARD IMAGE
IS IT A "C" ?
NO
SET "FIRST"≠0
SET FIRST OUTPUT ITEM LENGTH TO 14
YES
SET "FIRST"=0
SET FIRST OUTPUT ITEM LENGTH TO 24
STORE 1ST ITEM LENGTH IN "ADD7"
SAVE ADDRESS AT WHICH TO PUT ALPHANUMERIC IN 1ST OUTPUT ITEM
"ADD6"="ADD5"+1
GET col. 71 of INPUT CARD
STORE IT IN DECREMENT of "ADD3"
"TYPE" ?
=0
"CODE"=0
A2
≠0
GET Exclusion CODE (CARD column 70)
A1

CR-62021


A1
WHAT IS IN CARD col. 71 ?
N
D
Q
S
C
A or P
M
blank
other
"CODE"=6 "XTRA" ≠0
"CODE"=4
"CODE"=2 "XTRA" ≠0
"CODE"=7
"CODE"=5 "XTRA" ≠0
"CODE"=3 "XTRA" ≠0
"CODE"=1
"CODE"=0
A2
INDICATE 10TH INPUT WORD AS END of AlphANumeric
"TYPE"=0 ?
SET Profile Number TO zero
NO
STORE Profile Number IN "ADD1"
Get 1ST character of AlphANumeric INPUT Term
IS IT A WORD SEPARATOR ?
Set "KEY2" ≠0
NO
INPT

CR-62021


INPUT
Get (NEXT) 6 CHARACTERS of ALPHANUMERIC descriptor
STORE THEM IN CORRESPONDING PORTION of PRIMARY OUTPUT ITEM
"AWFUL" = 0 ?
NO
Yes
ARE ALL 6 CHARACTERS blANK ?
Yes
NO
PUT THEM INTO THE MQ REGISTER
SET INDICES: XR1 = CHARZ XR2 = 6 XR4 = ω AND "NCHAR" = 6
Get NEXT CHARACTER from MQ, PUT IN AC
clear
Yes
"NCHAR" > 0 ?
"NCHAR" = NCHAR - 1
CHAR
A3
No
SANE 1
ALL 60 CHAR. of CARD SCANNED ?
NO
INPUT
Yes
"AWFUL" ?
= 0
> 0
< 0
"KEY2" = 0 ?
NO
GATHER
XR1
Yes
"FIRST" = 0 ?
NO
Yes
SET "FIRST" ≠ 0
INDICATE 12th Thru 22nd INPUT WORDS HAVING THE Alphanumeric
EXCEPT
PREPARE TO CODE LAST ALPHANUMERIC WORD
A7

CR-62021

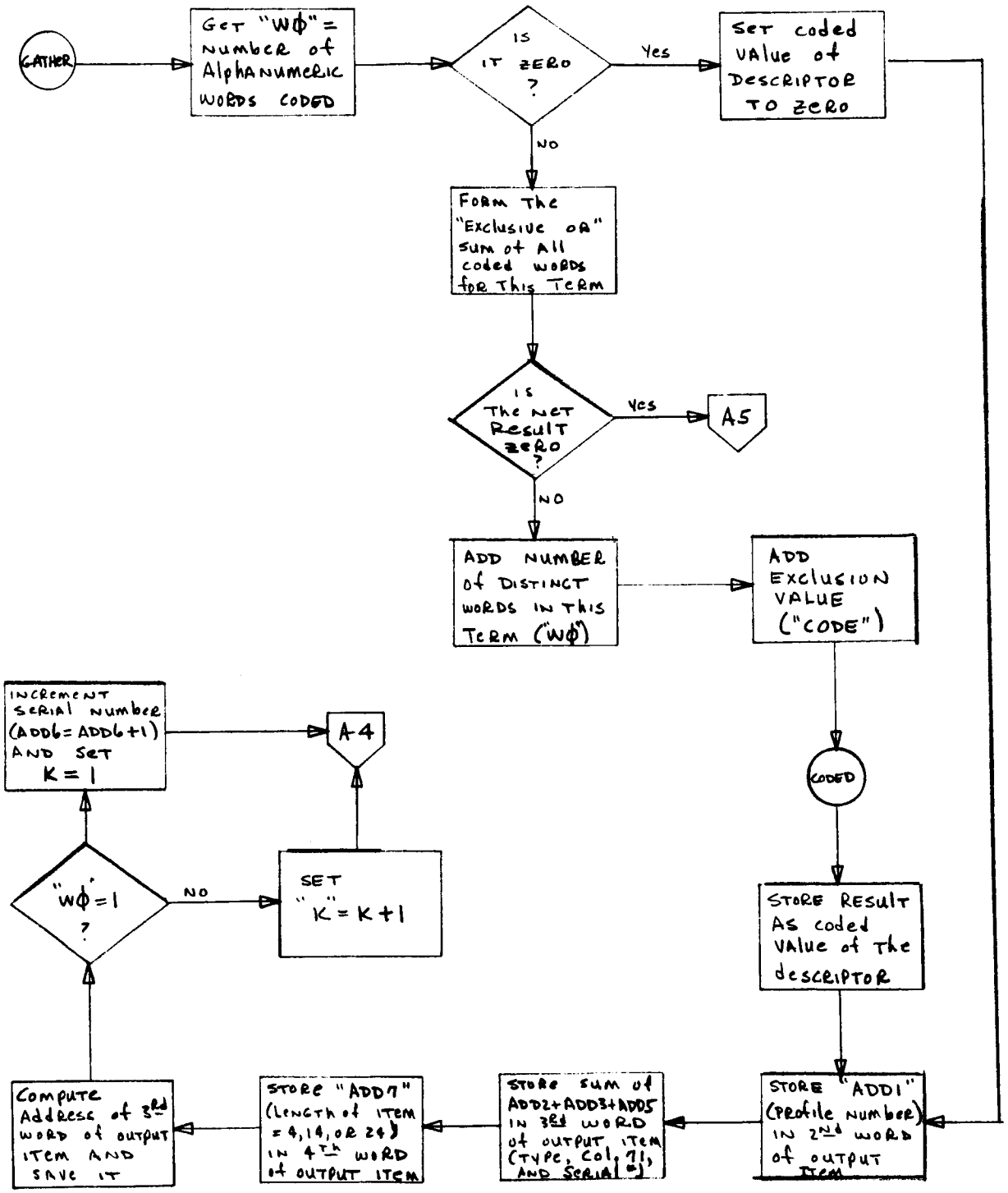
(A3) Table of Transfers Dependent on Current Character in Accumulator (AC)

If character is:		Go to
00	0	SAME
01	1	
02	2	
03	3	
04	4	
05	5	
06	6	
07	7	
10	8	
11	9	
12		GOOF
13	=	"
14	-	CHAR
15		GOOF
16		
17		
20		SAME
21	A	
22	B	
23	C	
24	D	
25	E	
26	F	
27	G	
30	H	
31	I	
32		GOOF
33	.	CHAR
34	)	GOOF
35		
36		
37		

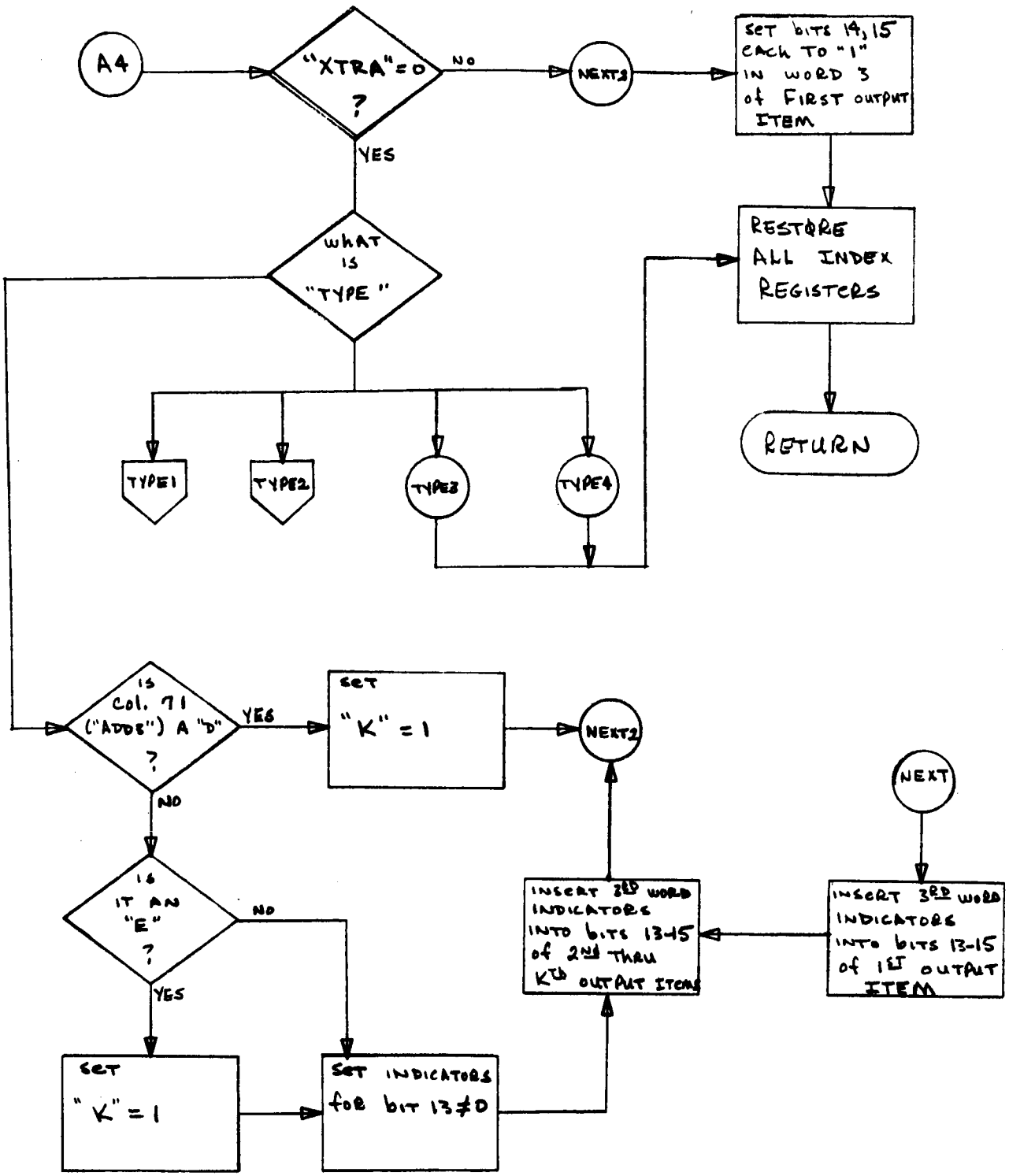
If character is:		Go to
40	-	CHAR
41	J	SAME
42	K	
43	L	
44	M	
45	N	
46	O	
47	P	
50	Q	
51	R	
52		GOOF
53	$	"
54	*	NEW
55		GOOF
56		
57		
60	Blank	NEW
61	/	"
62	S	SAME
63	T	
64	U	
65	V	
66	W	
67	X	
70	Y	
71	Z	
72		GOOF
73	,	NEW
74	(	GOOF
75		
76		
77		

Go to "SAME" if legitimate character of a word.
Go to "GOOF" if character is illegal.
Go to "CHAR" if character is to be ignored.
Go to "NEW" if character is a word separator.

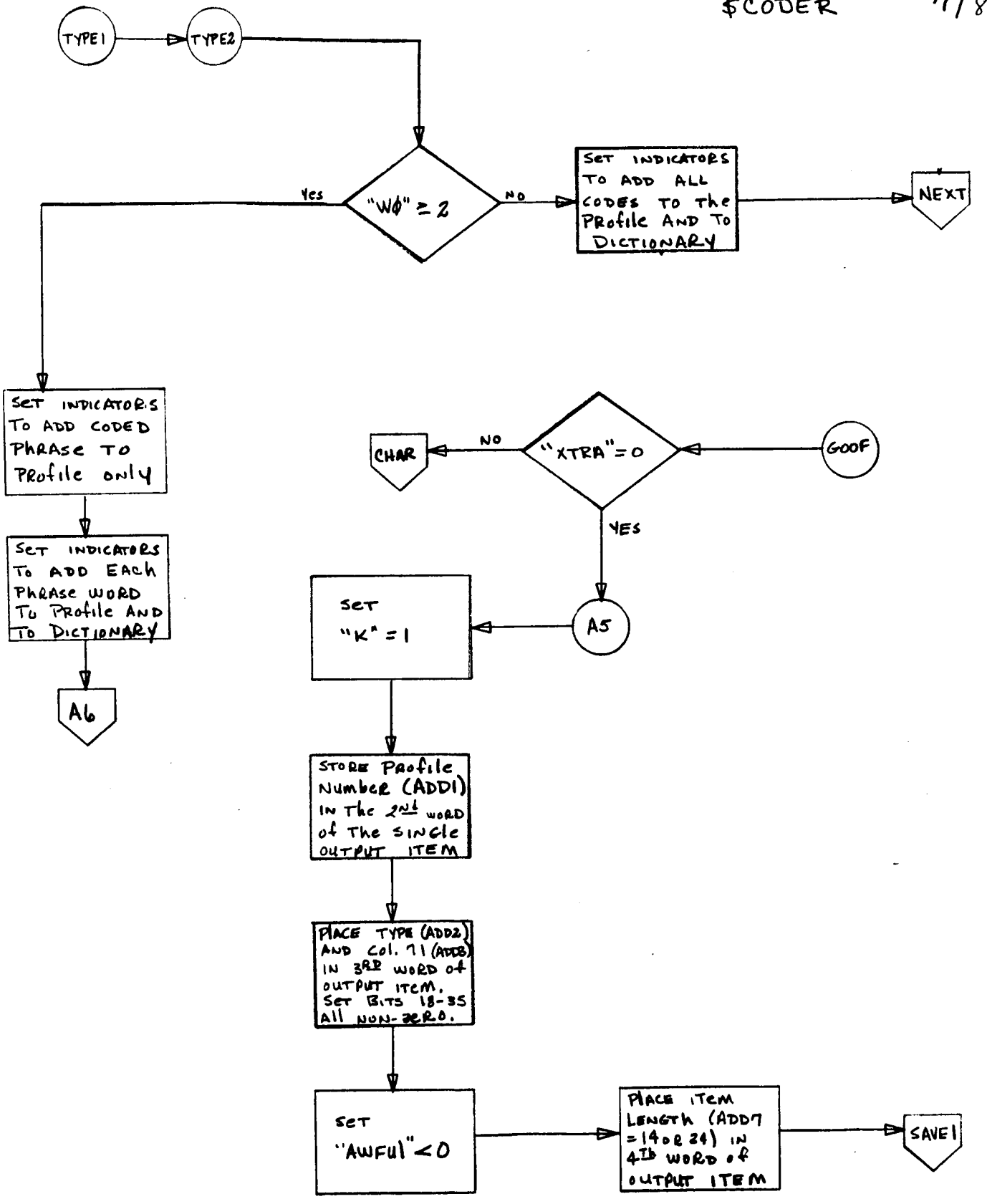
CR-62021


GATHER
Get "Wϕ" = Number of AlphaNumeric WORDS CODED
IS IT ZERO ?
Yes
Set coded VAlue of DESCRIPTOR TO ZERO
NO
FORM The "Exclusive OR" sum of All coded WORDS for This TERM
IS The NET Result ZERO ?
Yes
A5
NO
ADD NUMBER of DISTINCT WORDS IN THIS TERM ("Wϕ")
ADD Exclusion VALUE ("CODE")
CODED
STORE RESULT AS coded VAlue of the descriptor
STORE "ADD1" (Profile Number) IN 2nd WORD of output ITEM
STORE sum of ADD2+ADD3+ADD5 IN 3rd WORD of OUTPUT ITEM (TYPE, Col, TI, AND SERIAL #)
STORE "ADD7" (LENGTH of ITEM = 4, 14, OR 24) IN 4TH WORD of OUTPUT ITEM
Compute Address of 3rd WORD of output ITEM AND SAVE IT
"Wϕ" = 1 ?
NO
SET "K" = K+1
A4
INCREMENT SERIAL NUMBER (ADD6 = ADD6+1) AND SET K = 1

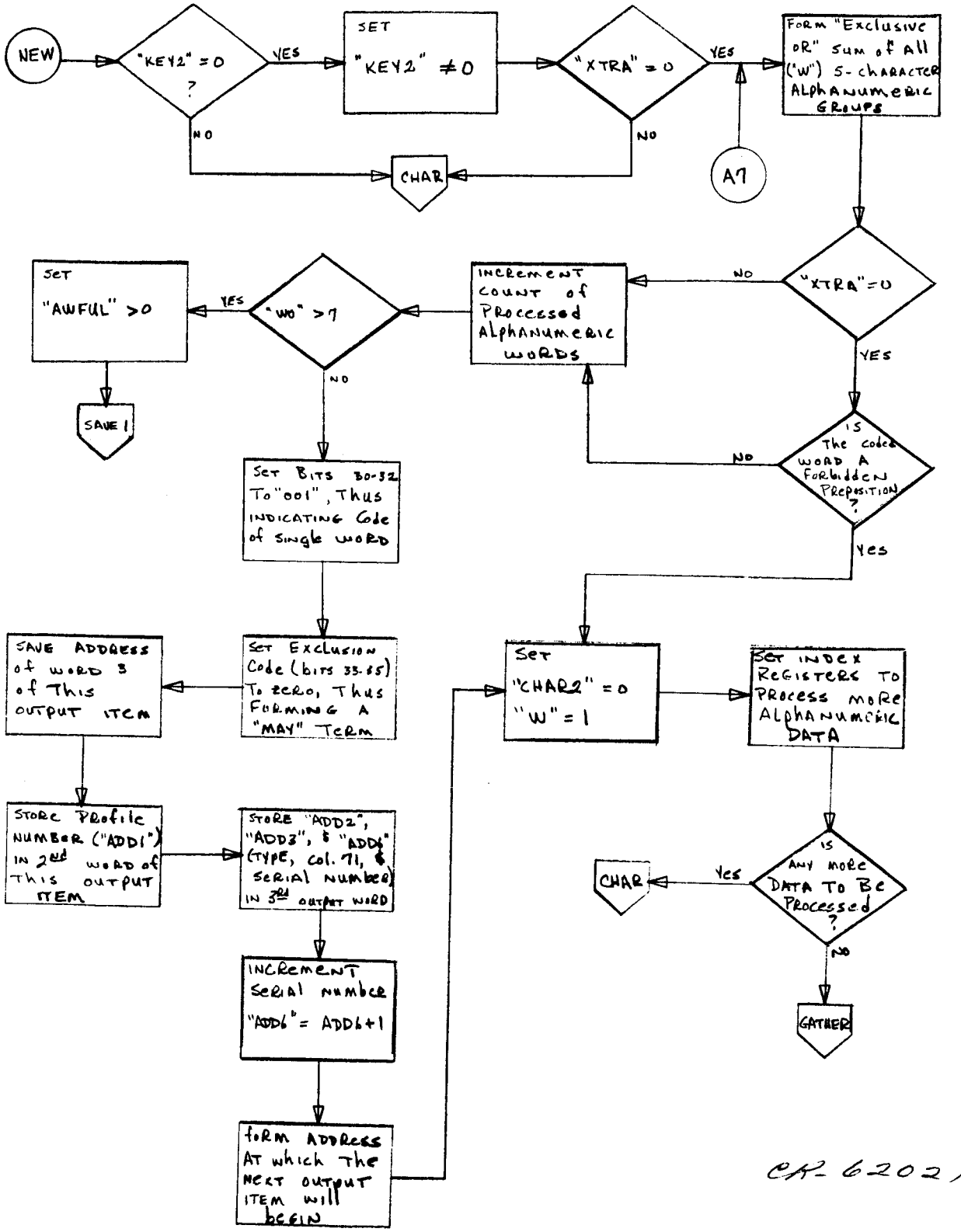
CR-62021


A4
"XTRA"=0 ?
NO
NEXT3
SET bits 14,15 EACH TO "1" IN WORD 3 of FIRST output ITEM
YES
WHAT IS "TYPE"
RESTORE ALL INDEX REGISTERS
RETURN
TYPE1
TYPE2
TYPE3
TYPE4
IS Col. 71 ("ADDE") A "D" ?
YES
SET "K" = 1
NEXT2
NO
IS IT AN "E" ?
NO
YES
SET "K" = 1
SET INDICATORS for bit 13≠0
INSERT 3RD WORD INDICATORS INTO bits 13-15 of 2ND THRU Kth output ITEMS
NEXT
INSERT 3RD WORD INDICATORS INTO bits 13-15 of 1ST output ITEM

CR-62021


TYPE1
TYPE2
"WØ" ≥ 2
Yes
NO
SET INDICATORS TO ADD ALL CODES TO the PROFILE AND TO DICTIONARY
NEXT
SET INDICATORS TO ADD CODED PHRASE TO PROFILE ONLY
SET INDICATORS TO ADD EACH PHRASE WORD TO PROFILE AND TO DICTIONARY
A6
CHAR
NO
"XTRA" = 0
GOOF
YES
SET "K" = 1
A5
STORE PROFILE NUMBER (ADD1) IN THE 2ND WORD of THE SINGLE OUTPUT ITEM
PLACE TYPE (ADD2) AND Col. 71 (ADD3) IN 3RD WORD of OUTPUT ITEM. SET BITS 18-35 All NON-ZERO.
SET "AWFUL" < 0
PLACE ITEM LENGTH (ADD7 = 14 OR 24) IN 4TH WORD of OUTPUT ITEM
SAVE1

CR-62021

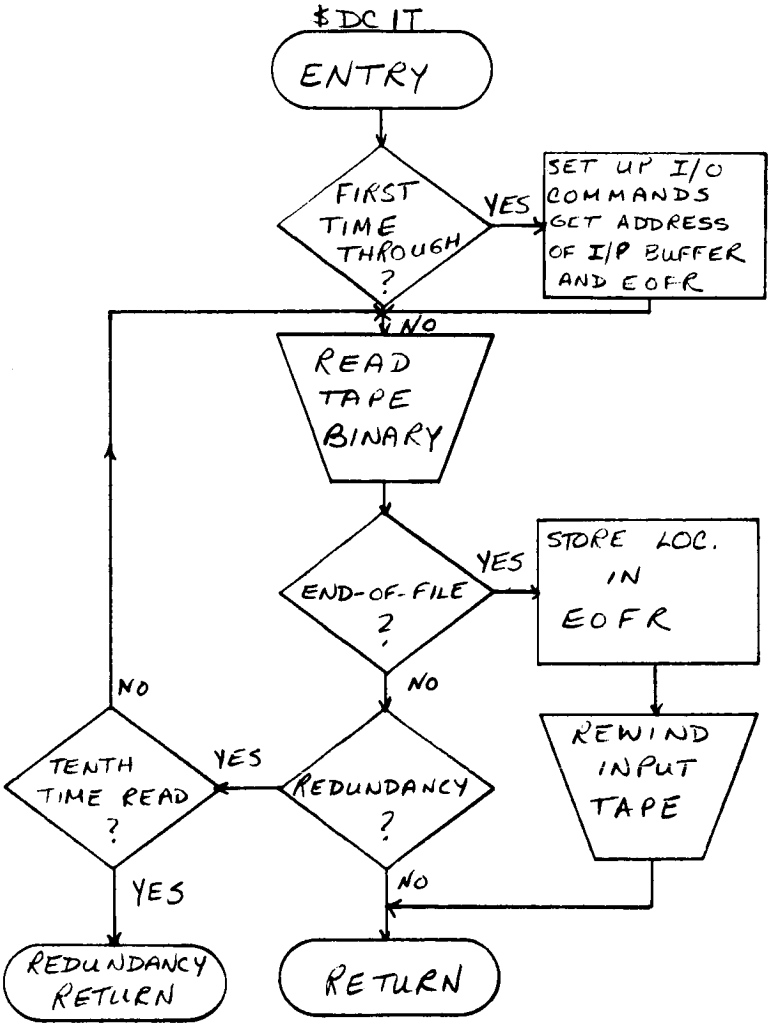

NEW
"KEY2" = 0 ?
YES
NO
SET "KEY2" ≠ 0
"XTRA" = 0
YES
NO
CHAR
A7
Form "Exclusive OR" sum of all ("W") 5-character ALPHANUMERIC GROUPS
"XTRA" = 0
NO
YES
INCREMENT COUNT of Processed ALPHANUMERIC WORDS
"W0" > 7
YES
NO
SET "AWFUL" > 0
SAVE 1
IS The coded WORD A Forbidden PREPOSITION ?
NO
yes
SET BITS 30-32 To "001", Thus INDICATING Code of single WORD
SET Exclusion Code (bits 33-35) To zero, Thus FORMING A "MAY" Term
SAVE ADDRESS of word 3 of This OUTPUT ITEM
STORE Profile NUMBER ("ADD1") IN 2nd WORD of This OUTPUT ITEM
STORE "ADD2", "ADD3", & "ADD4" (TYPE, col. 71, & SERIAL NUMBER) IN 3rd OUTPUT WORD
INCREMENT SERIAL number "ADD6" = ADD6+1
form ADDRESS AT which The NEXT OUTPUT ITEM will BEGIN
Set "CHAR2" = 0 "W" = 1
SET INDEX REGISTERS TO PROCESS MORE ALPHANUMERIC DATA
IS ANY MORE DATA TO BE Processed ?
yes
CHAR
NO
GATHER

CR-6202/

Subroutine DCIT

DCIT reads the sorted binary tape B containing the document descriptors. It is entered from subroutine PDTB. The read instructions are set up in its first entry. Upon reading EOF, the tape is rewound.

CR-62021


$DC IT
ENTRY
FIRST TIME THROUGH ?
YES
SET UP I/O COMMANDS GET ADDRESS OF I/P BUFFER AND EOFR
NO
READ TAPE BINARY
END-OF-FILE ?
YES
STORE LOC. IN EOFR
NO
REWIND INPUT TAPE
REDUNDANCY ?
YES
TENTH TIME READ ?
NO
YES
NO
REDUNDANCY RETURN
RETURN

CR-62021

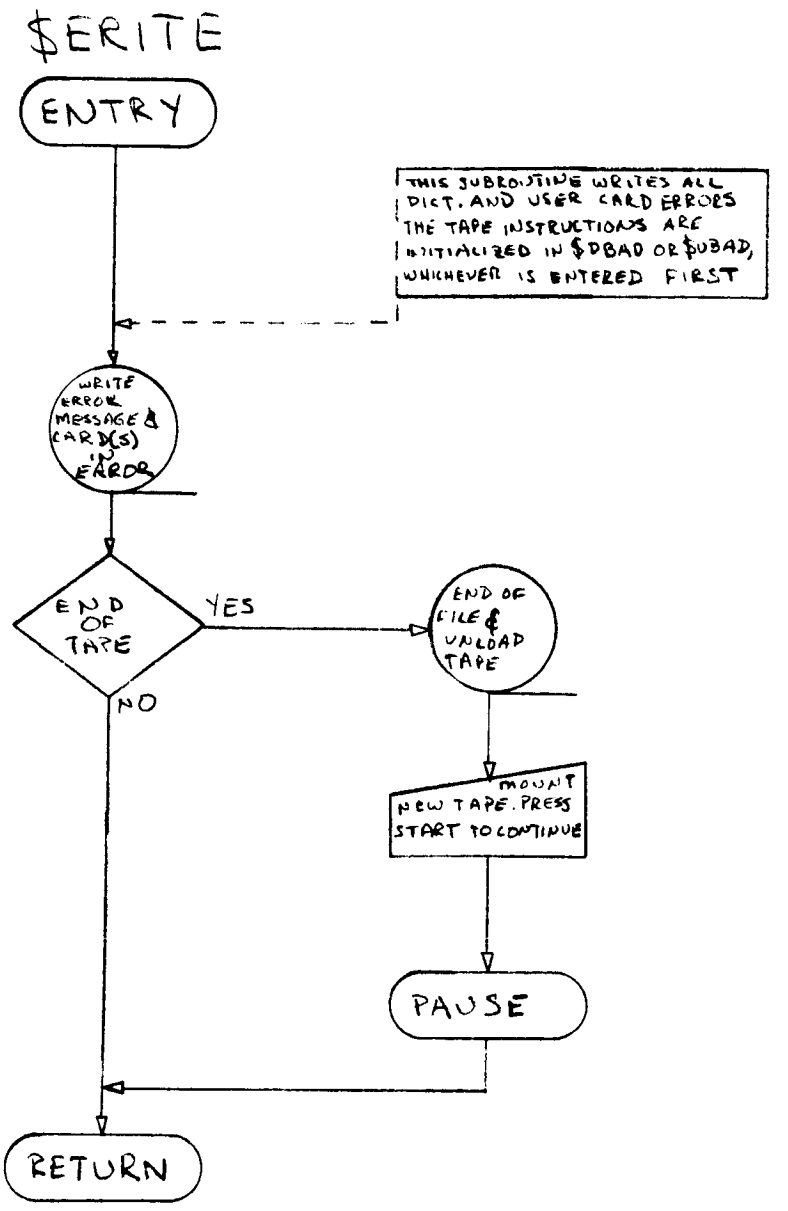
Subroutine ERITE

When an illegal card (user or vocabulary) is encountered, one of two subroutines (UBAD or DBAD) will TSX $ERITE, 4 to write the illegal card and the appropriate error message. The first time either $UBAD or $DBAD is entered, it will set up the output instructions in ERITE to write on the system output tape that has been specified in the tape assignment control card. Each time $UBAD or $DBAD is about to go to ERITE it will set up the RCH instruction in ERITE to write the illegal card or error message desired. After ERITE has written the BCD output record, it will return to $UBAD or $DBAD, whichever it came from, which in turn will go back to the subroutine processing user or vocabulary cards.

If an EOT is reached, an EOF will be written and the tape will be unloaded. The following message will be printed on line:

END OF TAPE WRITING cn. MOUNT NEW TAPE-PRESS START.
(c = channel, n = logical tape number)

CR-62021


$ERITE
ENTRY
THIS SUBROUTINE WRITES ALL DICT. AND USER CARD ERRORS THE TAPE INSTRUCTIONS ARE INITIALIZED IN $DBAD OR $UBAD, WHICHEVER IS ENTERED FIRST
WRITE ERROR MESSAGE & CARD(S) IN ERROR
END OF TAPE
YES
NO
END OF FILE & UNLOAD TAPE
MOUNT NEW TAPE. PRESS START TO CONTINUE
PAUSE
RETURN

CR-62021

Subroutine GDNRY

GDNRY processes all vocabulary input cards until a card with ENDICT in cols. 1-6 is encountered. Each card is picked up with a TSX $RDIN, 4. RDIN is a subroutine that reads one card from the input tape into area DCARD-DCARD+11.

Vocabulary input may be in one-card or two-card groups. If one card, col. 72 will be blank; if two, col. 72 of card 1 will contain a C and col. 72 of card 2 will be blank.

The legal cards with their punches in col. 71 are: add (A), delete (D), equate (E), and trouble term (*). If col. 71 does not contain any of the above, the entire card with an accompanying error message is written on the system output tape by subroutine DBAD. The card following a card with a C in col. 72 must match in col. 71, and in addition be blank in col. 72. Any irregularity will cause a rejection of the first card.

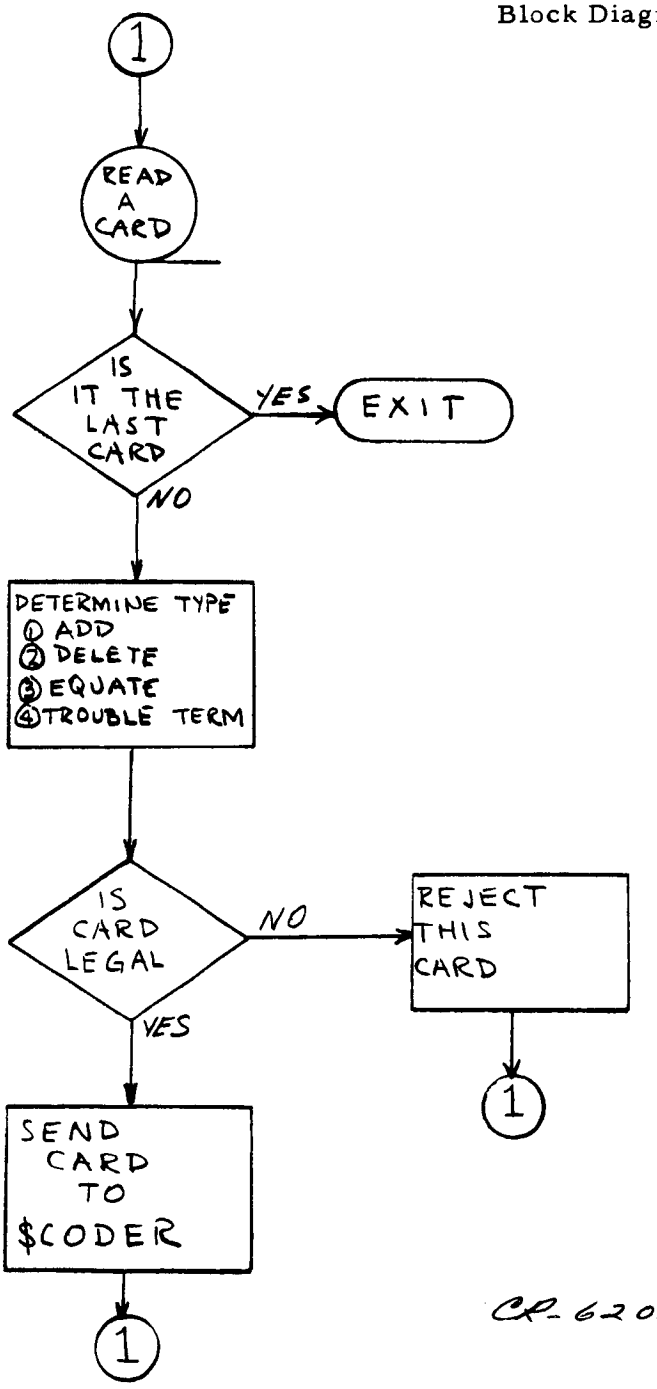
If GDNRY encounters an E-card, it must find a card with a T in col. 71 immediately after it. This contains the primary descriptor to which the phrase on the E-card will be equated. Any violation will cause rejection of the E-card or the E- and T-cards, depending upon the error.

Whenever a valid card group is read in, it will be sent to $CODER.

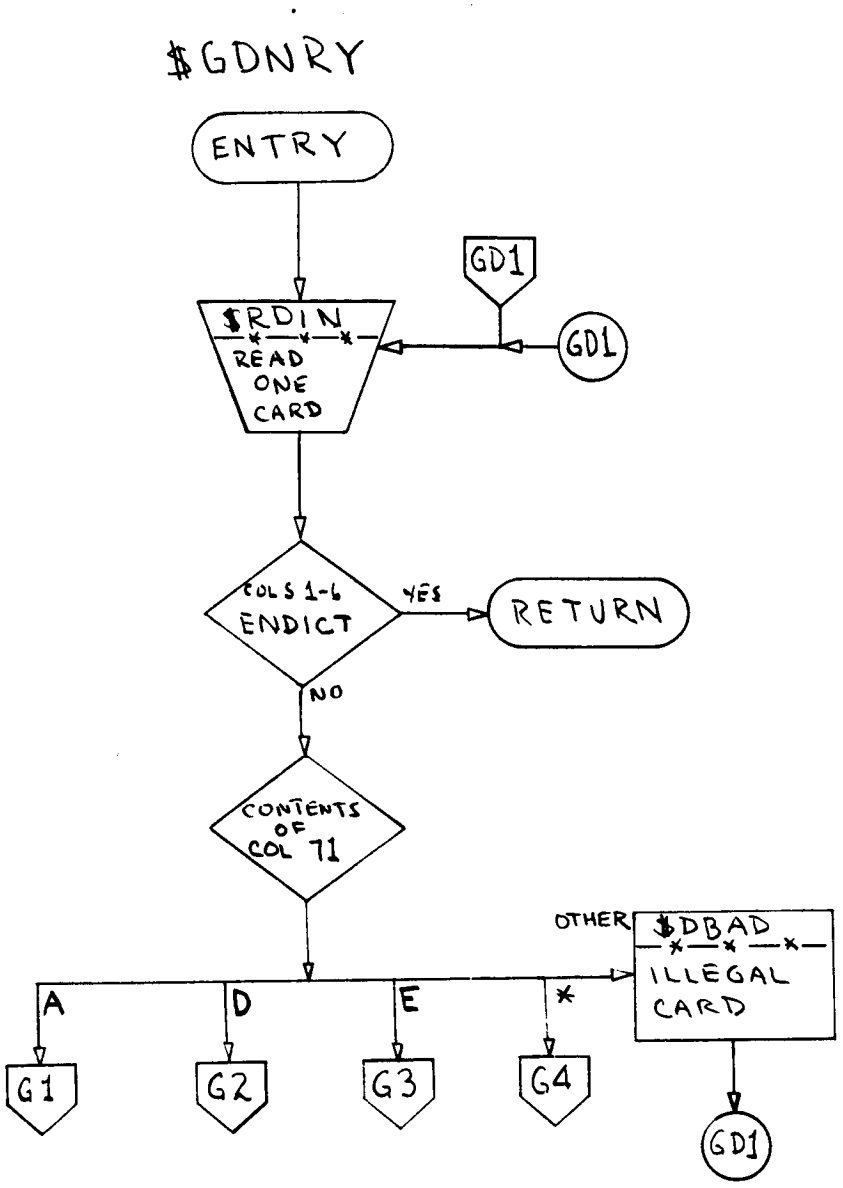
CR-62021

$GDNRY

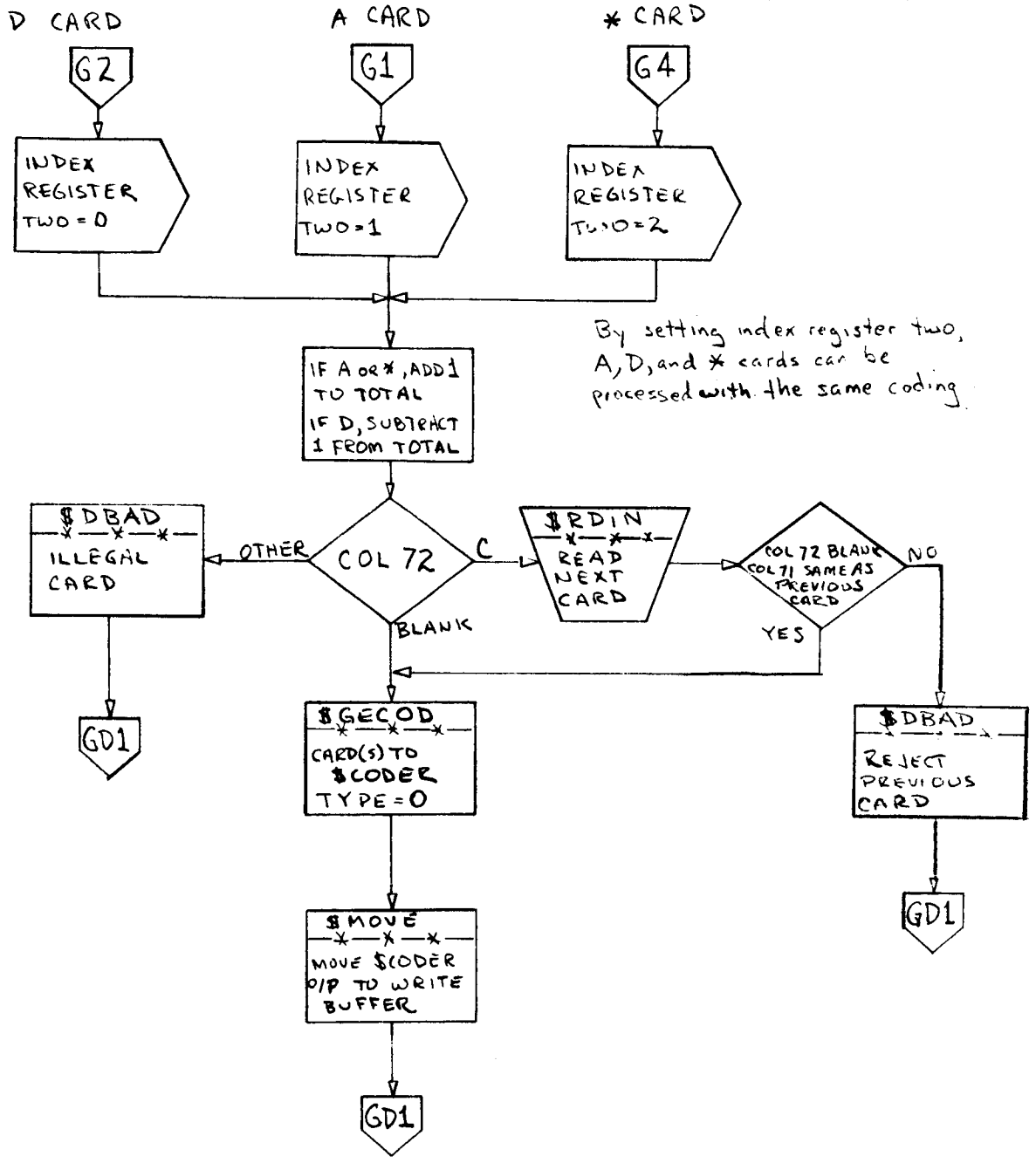
Block Diagram



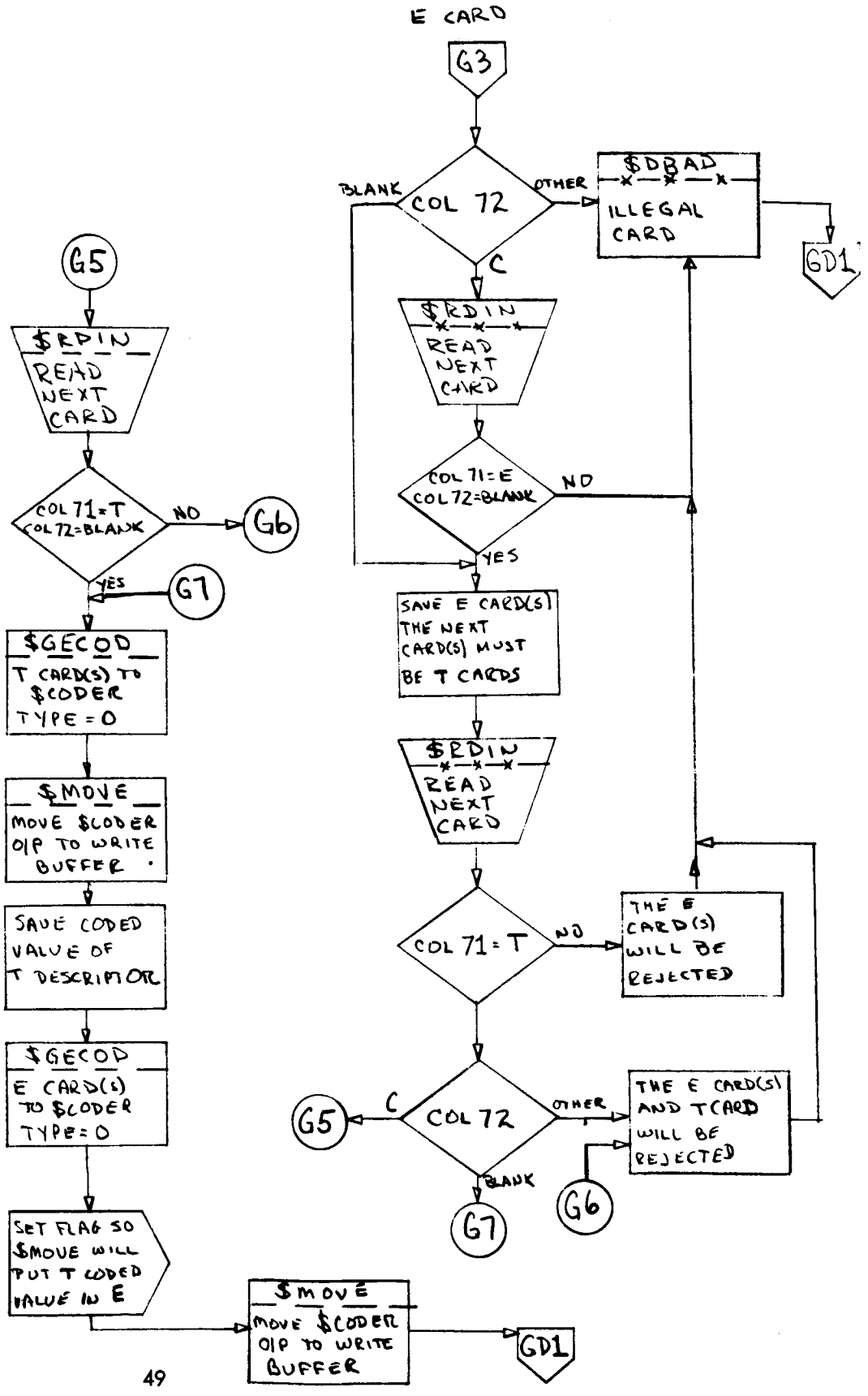
CR-62021


$GDNRY
ENTRY
GD1
$RDIN
READ ONE CARD
GD1
COLS 1-6 ENDICT
YES
RETURN
NO
CONTENTS OF COL 71
OTHER
$DBAD
ILLEGAL CARD
A
D
E
*
G1
G2
G3
G4
GD1

CR-62021


D CARD
A CARD
* CARD
G2
G1
G4
INDEX REGISTER TWO = 0
INDEX REGISTER TWO = 1
INDEX REGISTER TWO = 2
By setting index register two, A, D, and * cards can be processed with the same coding
IF A or *, ADD 1 TO TOTAL
IF D, SUBTRACT 1 FROM TOTAL
$DBAD
ILLEGAL CARD
OTHER
COL 72
C
$RDIN
READ NEXT CARD
COL 72 BLANK COL 71 SAME AS PREVIOUS CARD
NO
YES
BLANK
GD1
$GECOD
CARD(S) TO $CODER TYPE = 0
$DBAD
REJECT PREVIOUS CARD
GD1
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
GD1

CR-62021

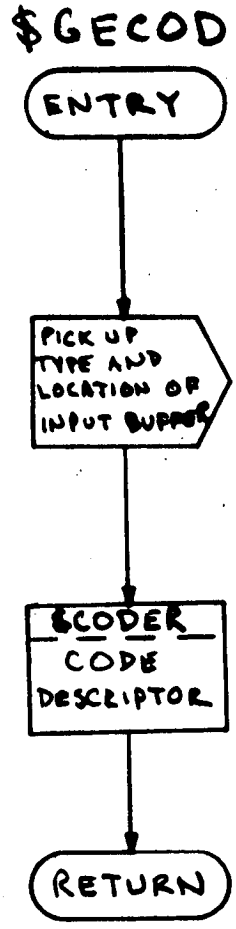

E CARD
G3
COL 72
BLANK
OTHER
C
$DBAD
ILLEGAL CARD
GD1
$RDIN
READ NEXT CARD
COL 71=E COL 72=BLANK
NO
YES
SAVE E CARD(S) THE NEXT CARD(S) MUST BE T CARDS
$RDIN
READ NEXT CARD
COL 71=T
NO
THE E CARD(S) WILL BE REJECTED
COL 72
C
G5
OTHER
THE E CARD(S) AND T CARD WILL BE REJECTED
BLANK
G7
G6
G5
G6
COL 71=T COL 72=BLANK
NO
YES
G7
$GECOD
T CARD(S) TO $CODER TYPE=0
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
SAVE CODED VALUE OF T DESCRIPTOR
$GECOD
E CARD(S) TO $CODER TYPE=0
SET FLAG SO $MOVE WILL PUT T CODED VALUE IN E
$MOVE
MOVE $CODER O/P TO WRITE BUFFER
GD1

CR-62021

Subroutine GECOD

Whenever a user, vocabulary, or document descriptor is to be sent to $CODER, a TSX $GECOD, 4 is executed. The calling sequence contains the location of the descriptor and the type. GECOD picks these up, sets them in the calling sequence $CODER expects, and then goes to $CODER.

CR-62021



CR-62021

(18 Sheet

Subroutine GTDOC

This subroutine reads the input document tape, and extracts the header and descriptors for each document. The header and all descriptors are written on tape A. Tape A also contains all vocabulary and user descriptors.

For each document, the following is done: The header is picked up and written on tape A. The next field (title field) is sent to $CODER. Five fields are skipped. The next field (personal author) is processed. Two more fields are skipped. Then the two remaining fields (contract number and descriptor fields) are processed. The last three fields handled by GTDOC may have more than one descriptor; each is separated by a word mark. Any field may be completely empty, in which case the field will contain one dollar sign. The dollar sign is the field separator.

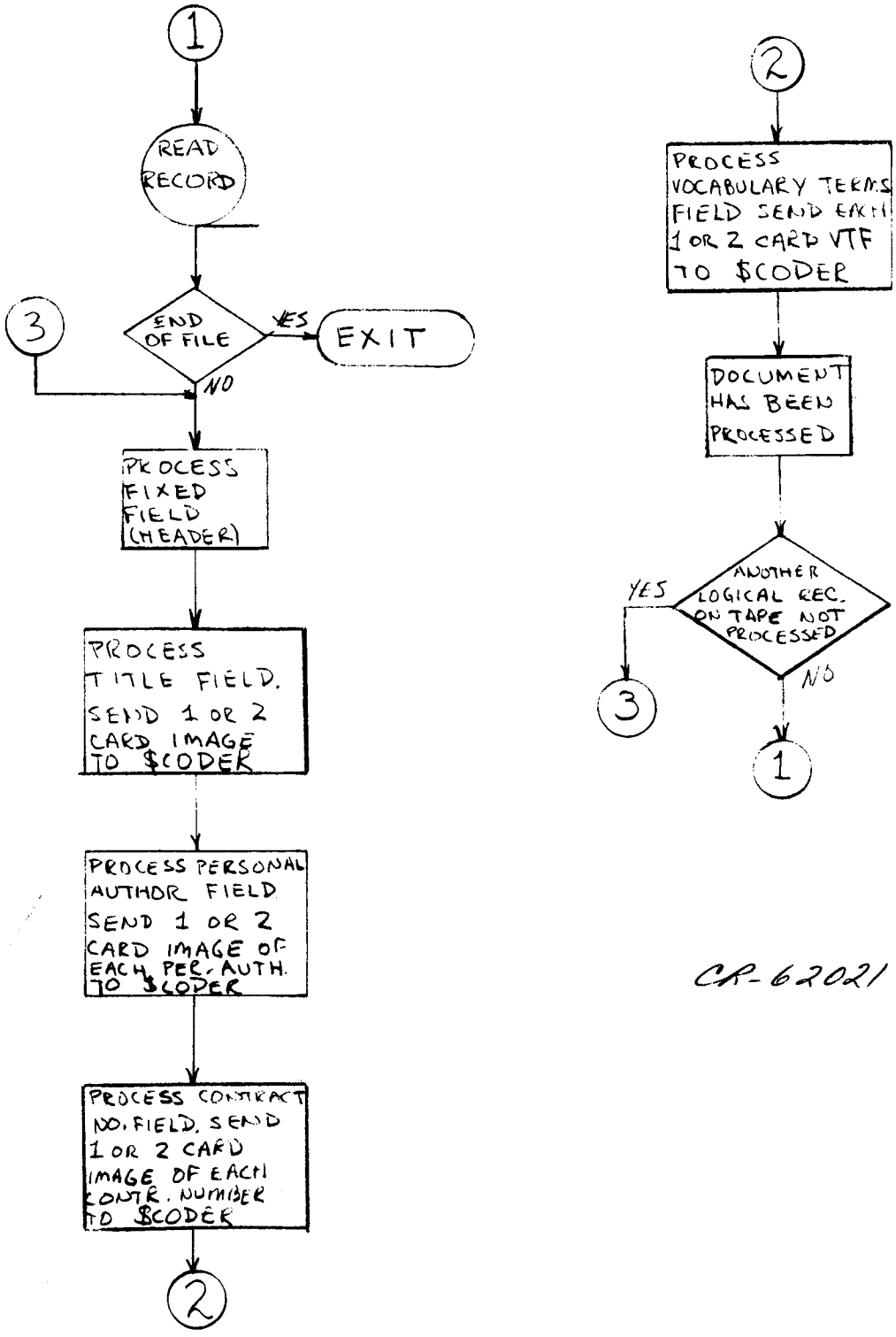
The document field called NOC may contain a price for the document, also indicated by a dollar sign, that is not a field separator. To avoid this false field separator, GTDOC utilizes the fields it wants from the beginning of the logical record, then skips the NOC field by counting backwards from the end of the logical record.

When EOF is encountered in reading the document tape, it is rewound.

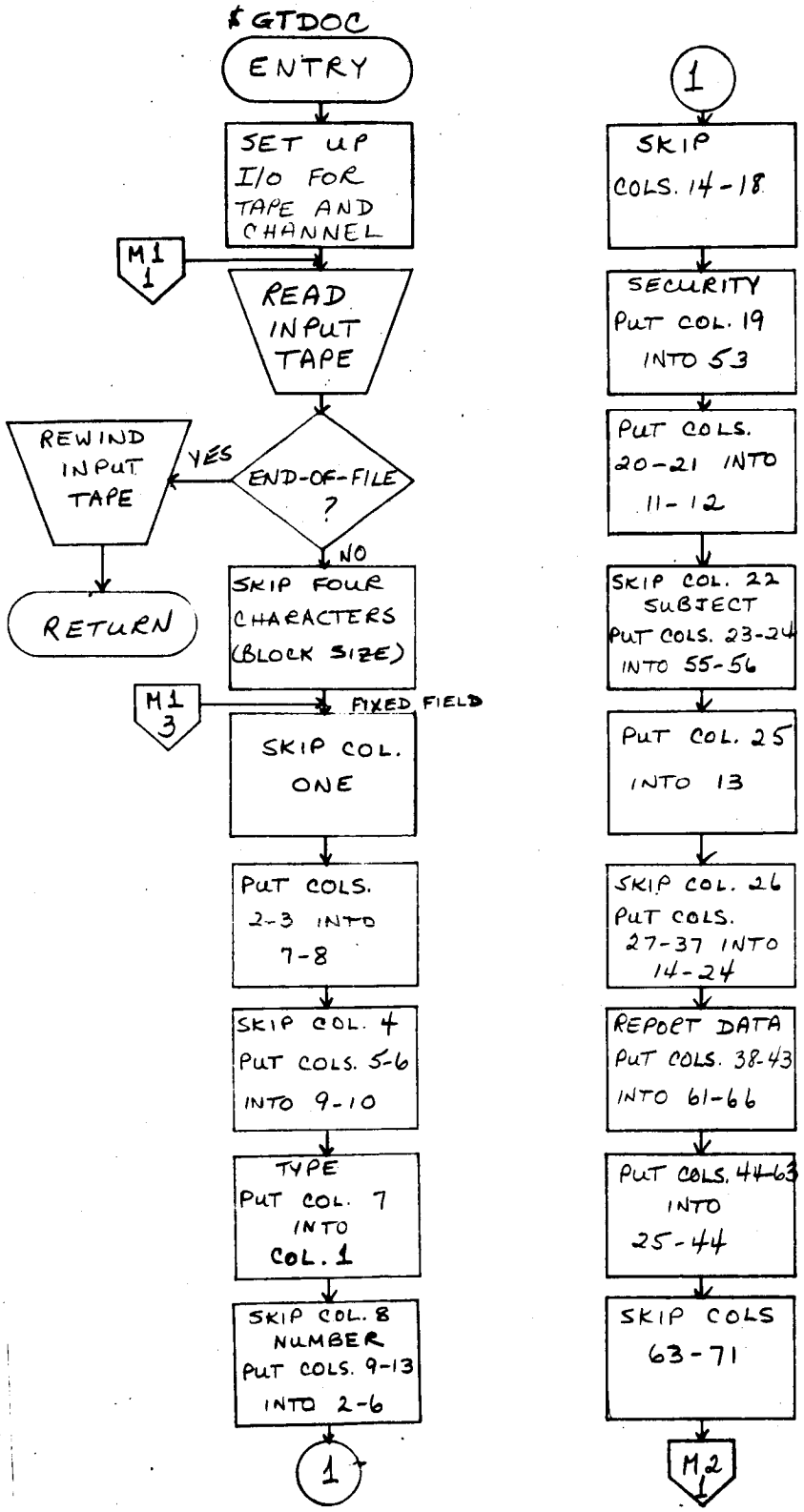
CR-62021

$GTDOC

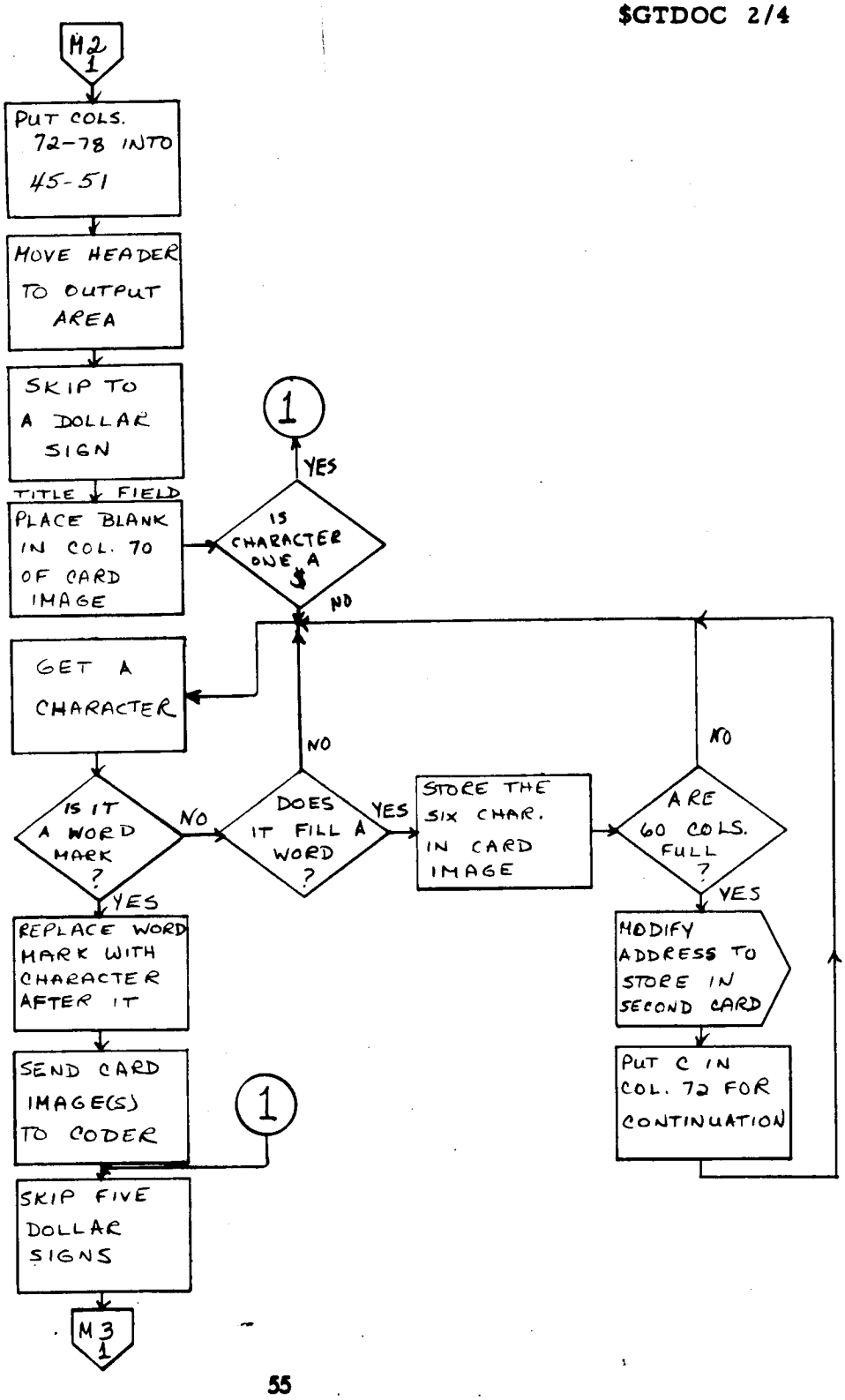
Block Diagram



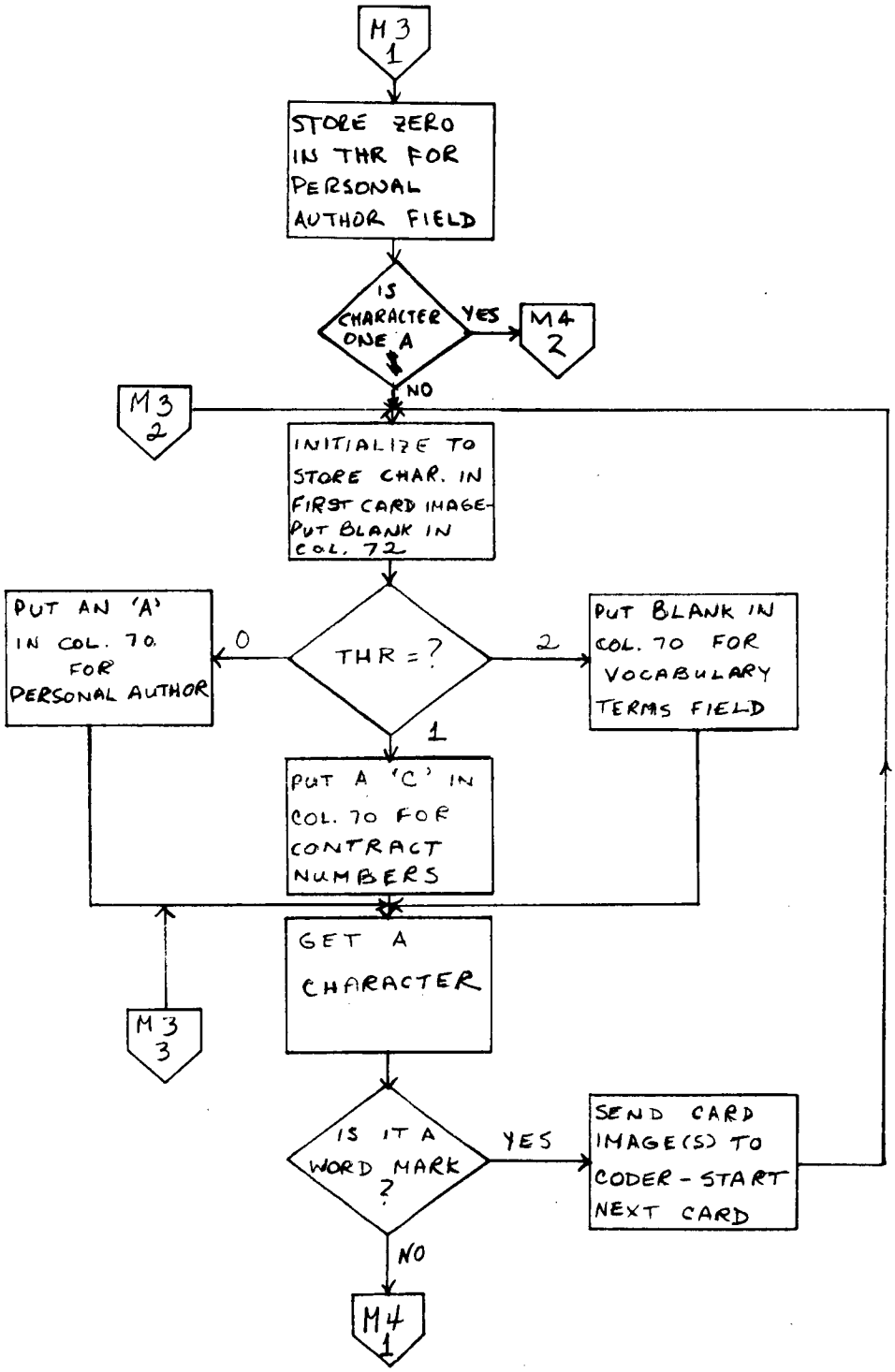
CR-62021


$GTDOC
ENTRY
SET UP I/O FOR TAPE AND CHANNEL
M1 1
READ INPUT TAPE
END-OF-FILE ?
YES
REWIND INPUT TAPE
RETURN
NO
SKIP FOUR CHARACTERS (BLOCK SIZE)
M1 3
FIXED FIELD
SKIP COL. ONE
PUT COLS. 2-3 INTO 7-8
SKIP COL. 4 PUT COLS. 5-6 INTO 9-10
TYPE PUT COL. 7 INTO COL. 1
SKIP COL. 8 NUMBER PUT COLS. 9-13 INTO 2-6
1
1
SKIP COLS. 14-18
SECURITY PUT COL. 19 INTO 53
PUT COLS. 20-21 INTO 11-12
SKIP COL. 22 SUBJECT PUT COLS. 23-24 INTO 55-56
PUT COL. 25 INTO 13
SKIP COL. 26 PUT COLS. 27-37 INTO 14-24
REPORT DATA PUT COLS. 38-43 INTO 61-66
PUT COLS. 44-63 INTO 25-44
SKIP COLS 63-71
M2 1

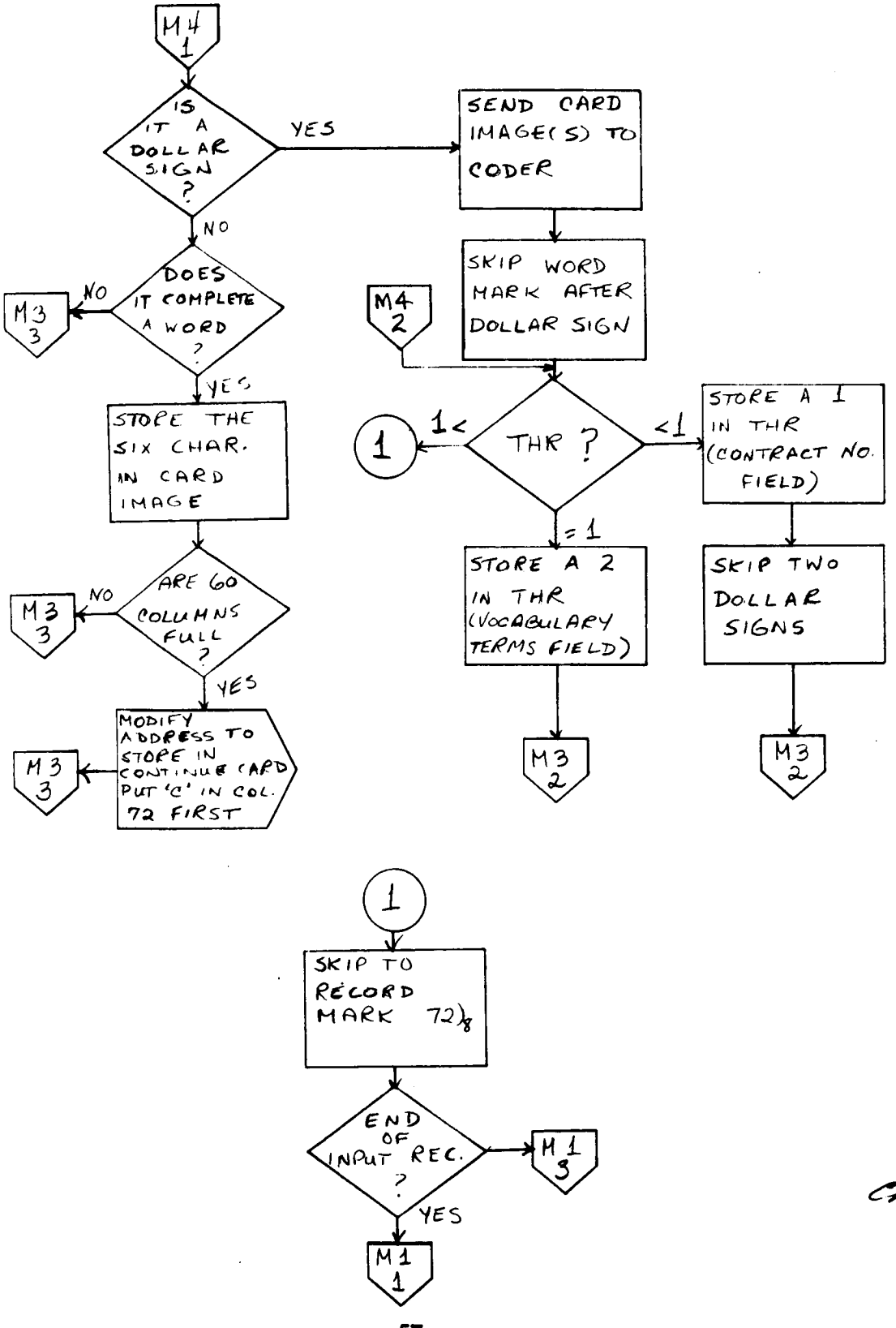
CR-62021


M2 1
PUT COLS. 72-78 INTO 45-51
MOVE HEADER TO OUTPUT AREA
SKIP TO A DOLLAR SIGN
TITLE FIELD
PLACE BLANK IN COL. 70 OF CARD IMAGE
1
YES
IS CHARACTER ONE A $
NO
GET A CHARACTER
NO
IS IT A WORD MARK ?
NO
DOES IT FILL A WORD ?
YES
STORE THE SIX CHAR. IN CARD IMAGE
NO
ARE 60 COLS. FULL ?
YES
YES
REPLACE WORD MARK WITH CHARACTER AFTER IT
MODIFY ADDRESS TO STORE IN SECOND CARD
SEND CARD IMAGE(S) TO CODER
1
PUT C IN COL. 72 FOR CONTINUATION
SKIP FIVE DOLLAR SIGNS
M3 1

CR-62021


M3 1
STORE ZERO IN THR FOR PERSONAL AUTHOR FIELD
IS CHARACTER ONE A
YES
M4 2
NO
M3 2
INITIALIZE TO STORE CHAR. IN FIRST CARD IMAGE-PUT BLANK IN COL. 72
PUT AN 'A' IN COL. 70 FOR PERSONAL AUTHOR
0
THR = ?
2
PUT BLANK IN COL. 70 FOR VOCABULARY TERMS FIELD
1
PUT A 'C' IN COL. 70 FOR CONTRACT NUMBERS
GET A CHARACTER
M3 3
IS IT A WORD MARK ?
YES
SEND CARD IMAGE(S) TO CODER - START NEXT CARD
NO
M4 1

CR-62021


M4 1
IS IT A DOLLAR SIGN ?
YES
SEND CARD IMAGE(S) TO CODER
NO
DOES IT COMPLETE A WORD ?
NO
M3 3
YES
STORE THE SIX CHAR. IN CARD IMAGE
ARE 60 COLUMNS FULL ?
NO
M3 3
YES
MODIFY ADDRESS TO STORE IN CONTINUE CARD PUT 'C' IN COL. 72 FIRST
M3 3
M4 2
SKIP WORD MARK AFTER DOLLAR SIGN
THR ?
1<
1
<1
STORE A 1 IN THR (CONTRACT NO. FIELD)
SKIP TWO DOLLAR SIGNS
M3 2
=1
STORE A 2 IN THR (VOCABULARY TERMS FIELD)
M3 2
1
SKIP TO RECORD MARK 72)8
END OF INPUT REC. ?
M1 3
YES
M1 1

CR-6.2021

Subroutine GTUSE

GTUSE processes all user input cards until a card with ENDUSR in cols. 1-6 is encountered. Each card is picked up with a TSX $RDIN, 4. Subroutine RDIN reads one card from the input tape into area DCARD-DCARD+11. The user number (cols. 61-66) is sent to subroutine RBLNK where all blanks are turned to zeros. This eliminates keypunching user numbers with leading zeros. Next, col. 71 is checked to see what kind of user card is being processed. The legal punches in col. 71 are "blank" (new user add), A (old user add), D (delete), R (report), or H (header). If the card does not contain any of the above, the entire card, with an accompanying error message, is written on the system output tape by subroutine UBAD as an illegal card. The next card is then read.

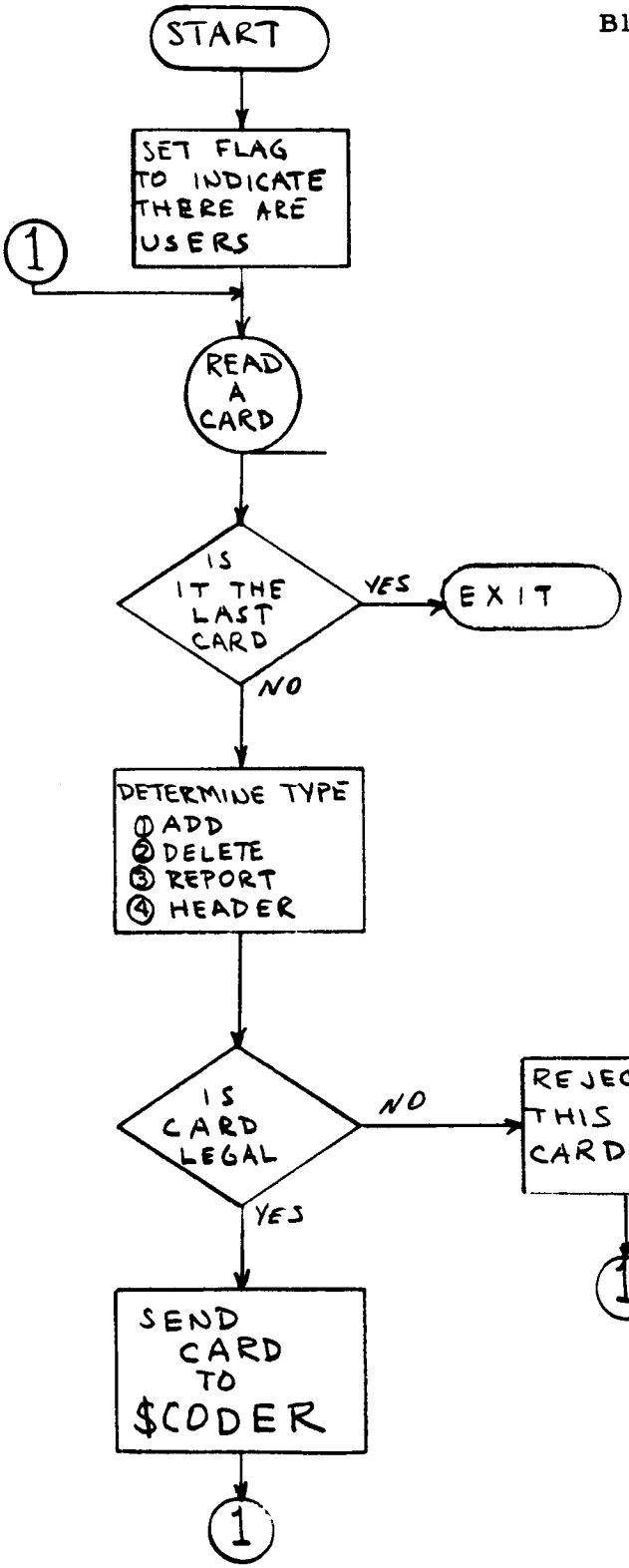
Delete cards may be of two types. One type deletes a user descriptor and the second deletes an entire user profile. The latter is denoted by a U punched in col. 72. With the exception of report cards, user input may be in one-card or two-card groups. If one card, col. 72 will be blank. If two cards, col. 72 of card 1 will contain a C and col. 72 of card 2 will be blank. Any other punch is illegal. In two-card groups, col. 71 of both cards must agree.

All add and delete cards are sent to $CODER and then written on tape A, using subroutines GECOD, MOVE, and WTAPA. Tape A contains vocabulary and document input, in addition to user input. Report type cards and header cards are not sent to $CODER, but are written directly on tape A by $WTAPA.

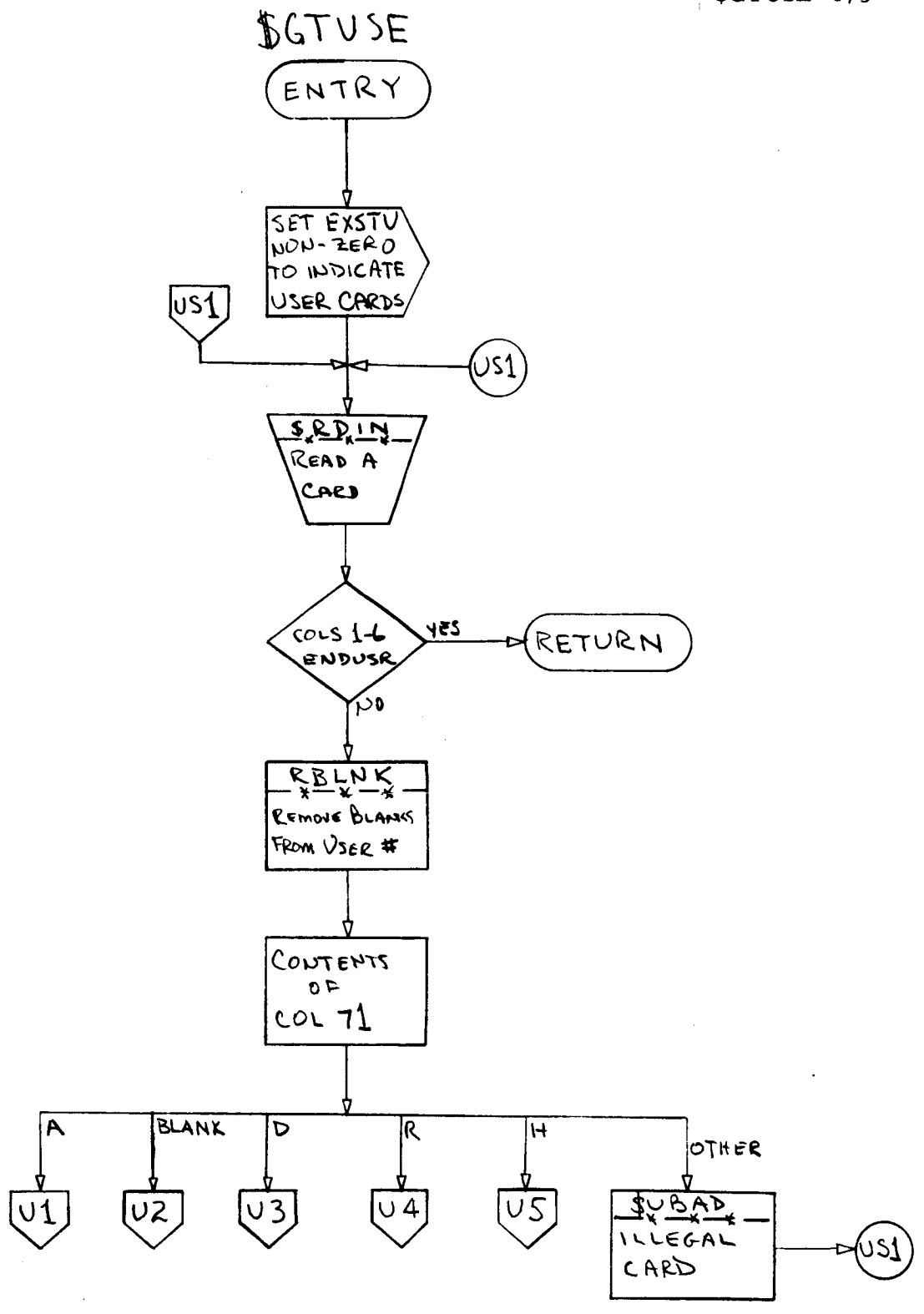
Header cards are given a serialization number of 000000)8 to place them before any descriptors for that user. Delete profile and report cards are given very high numbers to place them after all add and delete descriptor cards. Their serial numbers are 777774)8 and 777776)8, respectively. The serial numbers for adds and deletes are assigned by $CODER, and will always be between 000001)8 and 777773)8.

CR-62021

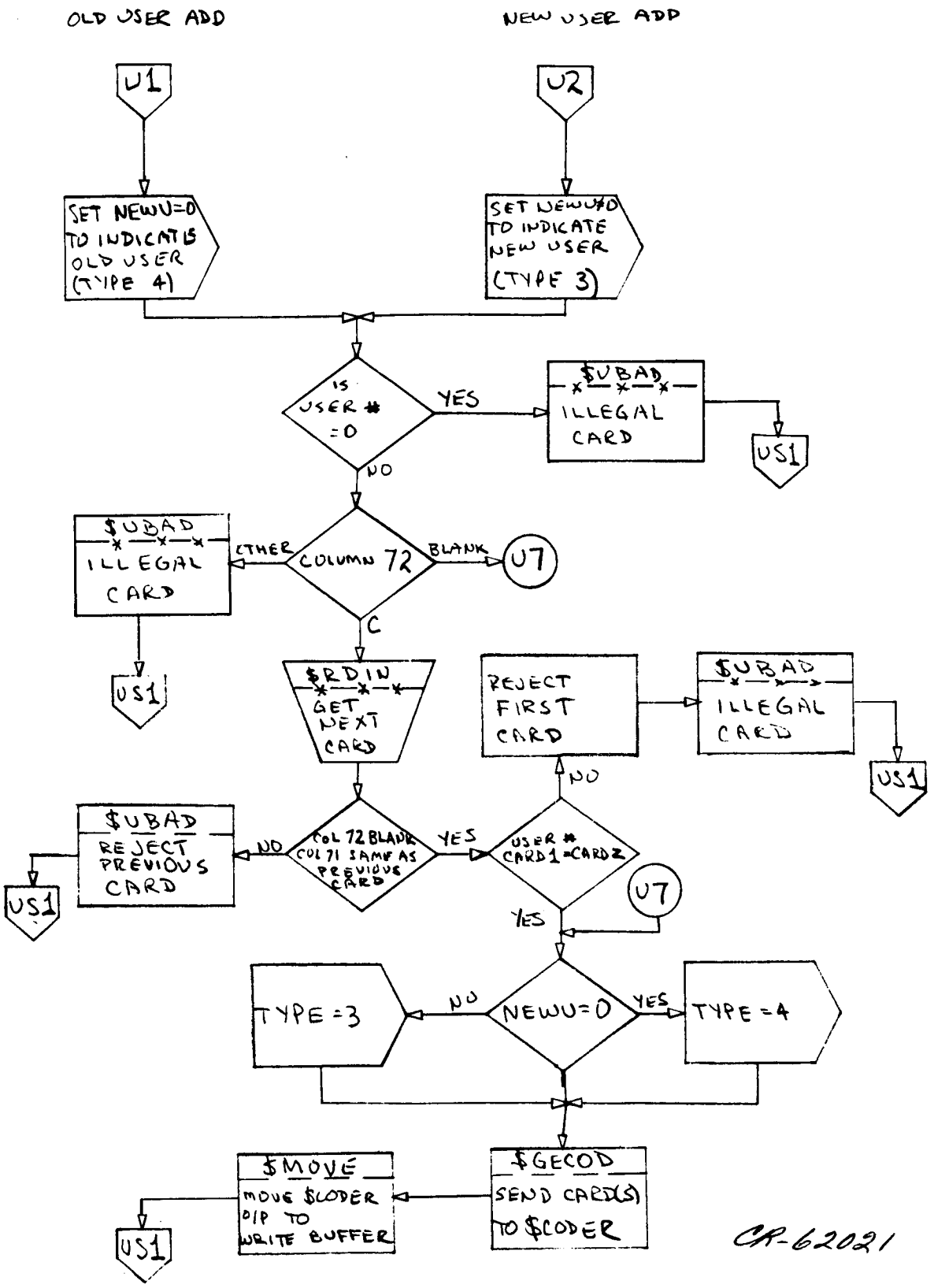
Block Diagram

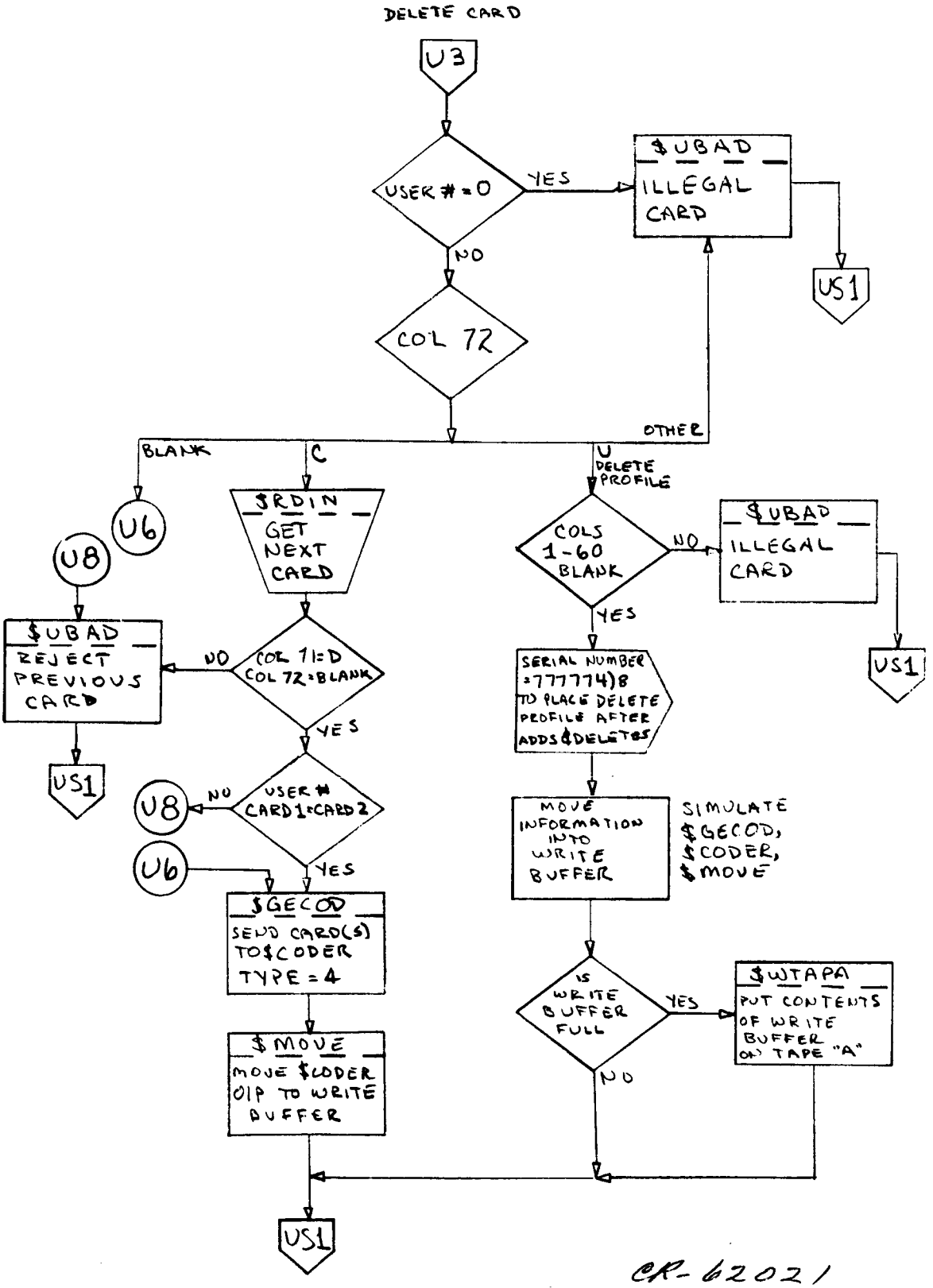


CR-62021


$GTUSE
ENTRY
SET EXSTU NON-ZERO TO INDICATE USER CARDS
US1
US1
$RDIN
READ A CARD
COLS 1-6 ENDUSR
YES
RETURN
NO
RBLNK
REMOVE BLANKS FROM USER #
CONTENTS OF COL 71
A
BLANK
D
R
H
OTHER
U1
U2
U3
U4
U5
$UBAD
ILLEGAL CARD
US1

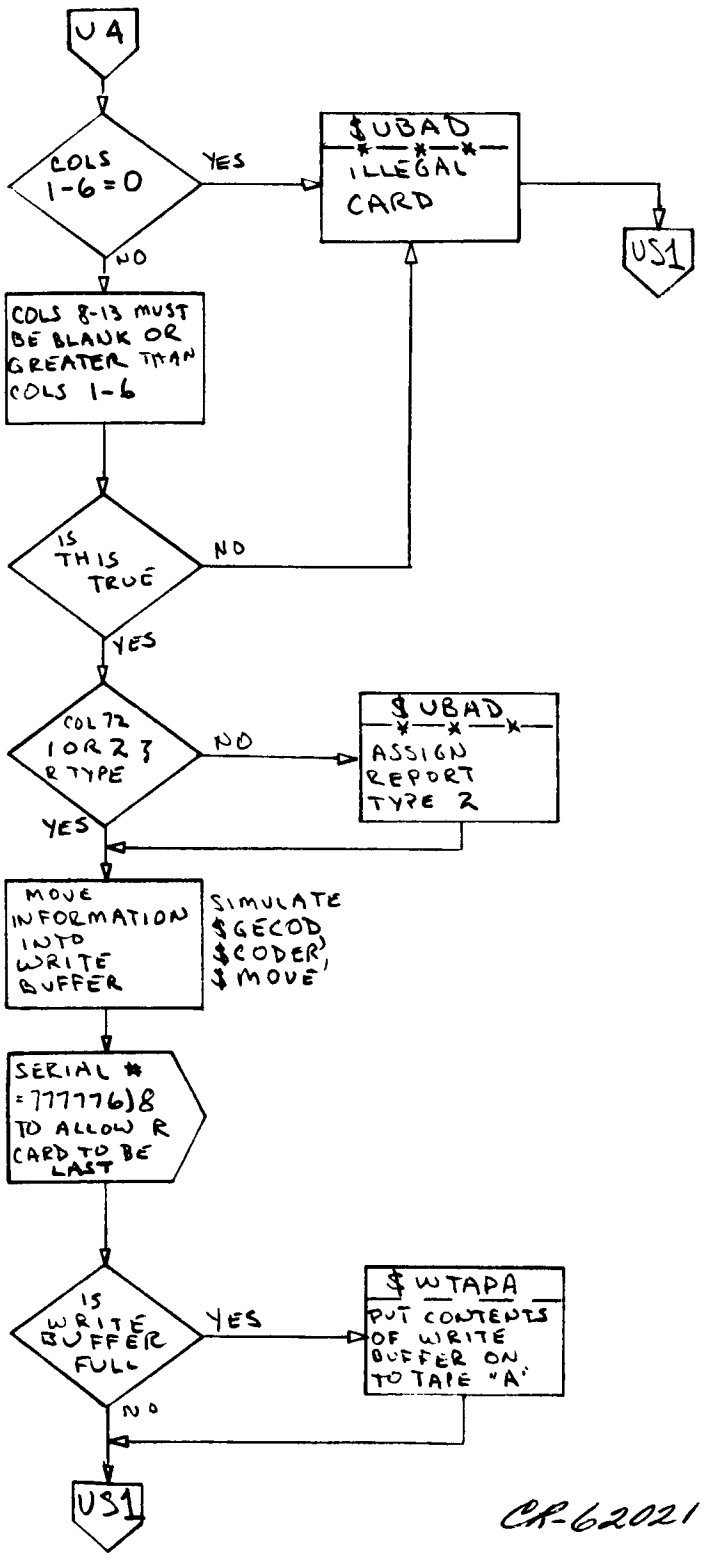
CR-62021


OLD USER ADD
NEW USER ADD
U1
U2
SET NEWU=0 TO INDICATE OLD USER (TYPE 4)
SET NEWU≠0 TO INDICATE NEW USER (TYPE 3)
IS USER # =0
YES
NO
$UBAD ILLEGAL CARD
US1
$UBAD ILLEGAL CARD
OTHER
COLUMN 72
BLANK
U7
C
US1
$RDIN GET NEXT CARD
REJECT FIRST CARD
$UBAD ILLEGAL CARD
US1
NO
$UBAD REJECT PREVIOUS CARD
NO
COL 72 BLANK COL 71 SAME AS PREVIOUS CARD
YES
USER # CARD 1 = CARD 2
U7
US1
YES
TYPE =3
NO
NEWU=0
YES
TYPE =4
$MOVE MOVE $LODER O/P TO WRITE BUFFER
$GECOD SEND CARD(S) TO $LODER
US1
CR-62021

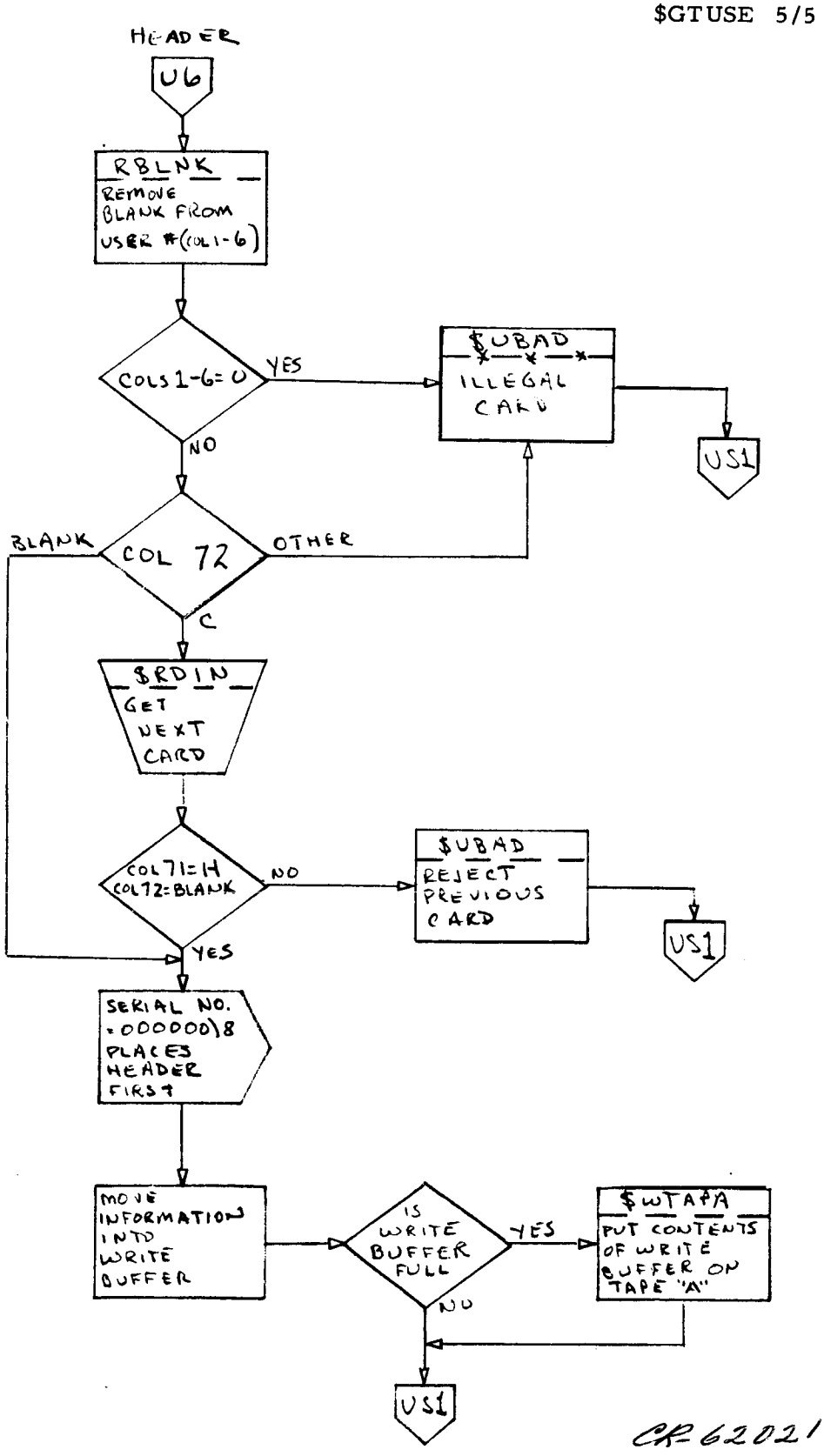

DELETE CARD
U3
USER # = 0
YES
$UBAD
ILLEGAL CARD
US1
NO
COL 72
OTHER
BLANK
C
U
DELETE PROFILE
U6
U8
$RDIN
GET NEXT CARD
COLS 1-60 BLANK
NO
$UBAD
ILLEGAL CARD
US1
YES
$UBAD
REJECT PREVIOUS CARD
US1
NO
COL 1 = D
COL 72 = BLANK
YES
SERIAL NUMBER = 777774)8
TO PLACE DELETE PROFILE AFTER ADDS & DELETES
U8
NO
USER # CARD 1 = CARD 2
YES
U6
MOVE INFORMATION INTO WRITE BUFFER
SIMULATE $GECOD, $CODER, $MOVE
$GECOD
SEND CARD(S) TO $CODER TYPE = 4
IS WRITE BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TAPE "A"
NO
$MOVE
MOVE $LODER O/P TO WRITE BUFFER
US1

CR-62021

REPORT



CR-62021

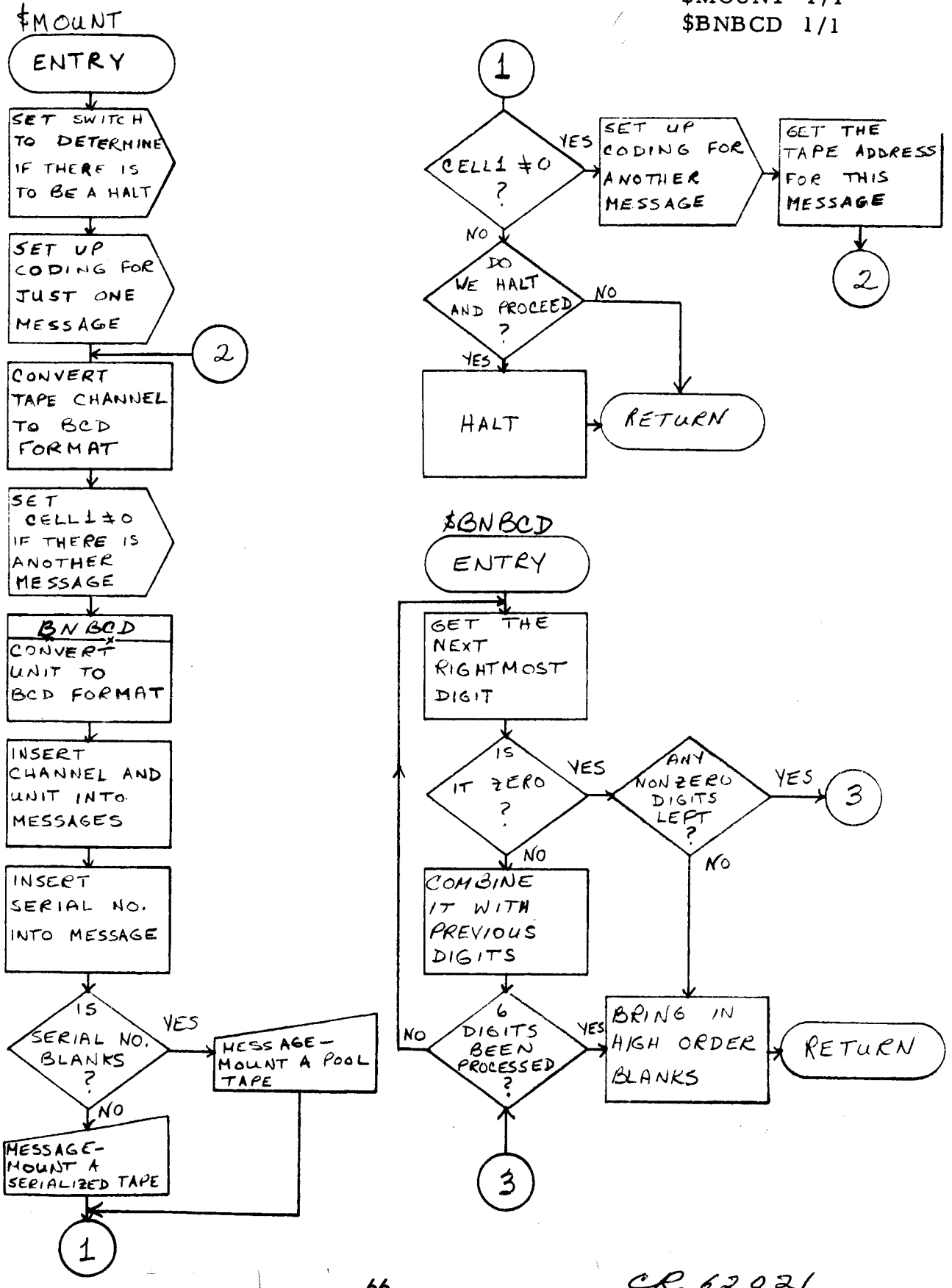

HEADER
U6
RBLNK
REMOVE BLANK FROM USER #(COL 1-6)
COLS 1-6 = 0
YES
$UBAD
ILLEGAL CARD
US1
NO
BLANK
COL 72
OTHER
C
$RDIN
GET NEXT CARD
COL 71 = H
COL 72 = BLANK
NO
$UBAD
REJECT PREVIOUS CARD
US1
YES
SERIAL NO. = 000000\8 PLACES HEADER FIRST
MOVE INFORMATION INTO WRITE BUFFER
IS WRITE BUFFER FULL
YES
$WTAPA
PUT CONTENTS OF WRITE BUFFER ON TAPE "A"
NO
US1

CR 62021

Subroutine MOUNT - BNBCD

This subroutine prints messages on the online printer when a tape has to be mounted. If the tape has a serial number, this will be given in the message; if not, the message will ask that a pool tape be mounted. This subroutine has within it a routine to convert an unsigned binary integer to a BCD integer with leading blanks.

CR-62021


$MOUNT
ENTRY
SET SWITCH TO DETERMINE IF THERE IS TO BE A HALT
SET UP CODING FOR JUST ONE MESSAGE
2
CONVERT TAPE CHANNEL TO BCD FORMAT
SET CELL1 ≠0 IF THERE IS ANOTHER MESSAGE
BNBCD
CONVERT UNIT TO BCD FORMAT
INSERT CHANNEL AND UNIT INTO MESSAGES
INSERT SERIAL NO. INTO MESSAGE
IS SERIAL NO. BLANKS ?
YES
MESSAGE-MOUNT A POOL TAPE
NO
MESSAGE-MOUNT A SERIALIZED TAPE
1
1
CELL1 ≠0 ?
YES
SET UP CODING FOR ANOTHER MESSAGE
GET THE TAPE ADDRESS FOR THIS MESSAGE
2
NO
DO WE HALT AND PROCEED ?
NO
YES
HALT
RETURN
$BNBCD
ENTRY
GET THE NEXT RIGHTMOST DIGIT
IS IT ZERO ?
YES
ANY NONZERO DIGITS LEFT ?
YES
3
NO
NO
COMBINE IT WITH PREVIOUS DIGITS
6 DIGITS BEEN PROCESSED ?
NO
YES
BRING IN HIGH ORDER BLANKS
RETURN
3

CR-62021

Subroutine MOVE

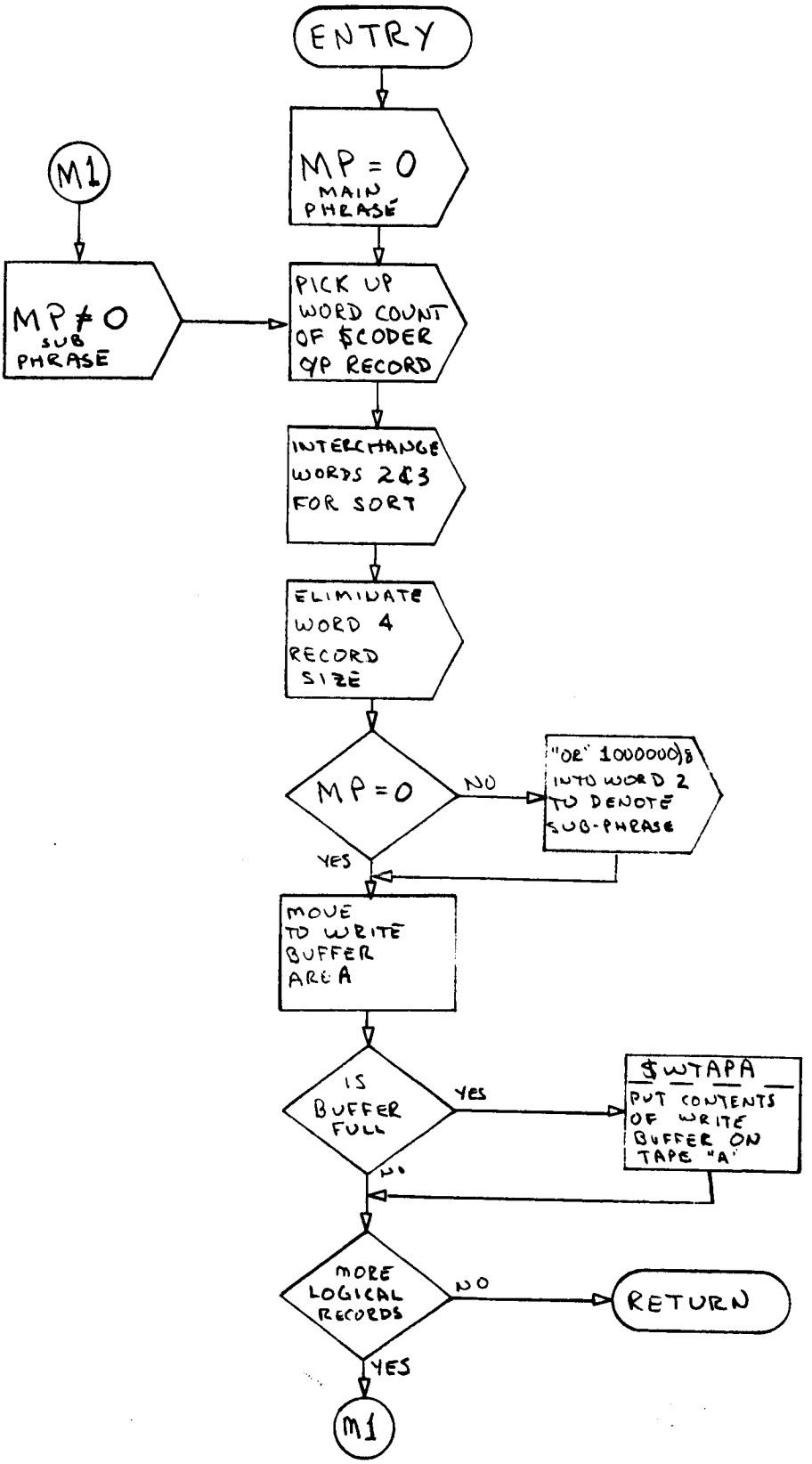
MOVE is entered after every TSX $CODER, 4. It picks up the logical records put out by $CODER (as few as one, as many as eight) and places them in BLOCK, preparing them to be written on tape A. After each logical record is moved into the BLOCK area, a check is made to see whether or not six logical records are in BLOCK. If so, MOVE will TSX $WTAPA, 4 to write out these six. When it returns it will process the remaining records produced by $CODER.

The first word of every main phrase from $CODER will be masked to contain a "1" in bit 17. All subphrases will be masked so their first words will contain a "0" in this same position.

Special attention is given to vocabulary equate cards. If an E-card has come from $CODER, the coded value of its accompanying T-card will be placed in word 24 of the logical record containing the E-card main phrase.

CR-62021

$MOVE



CR-62021

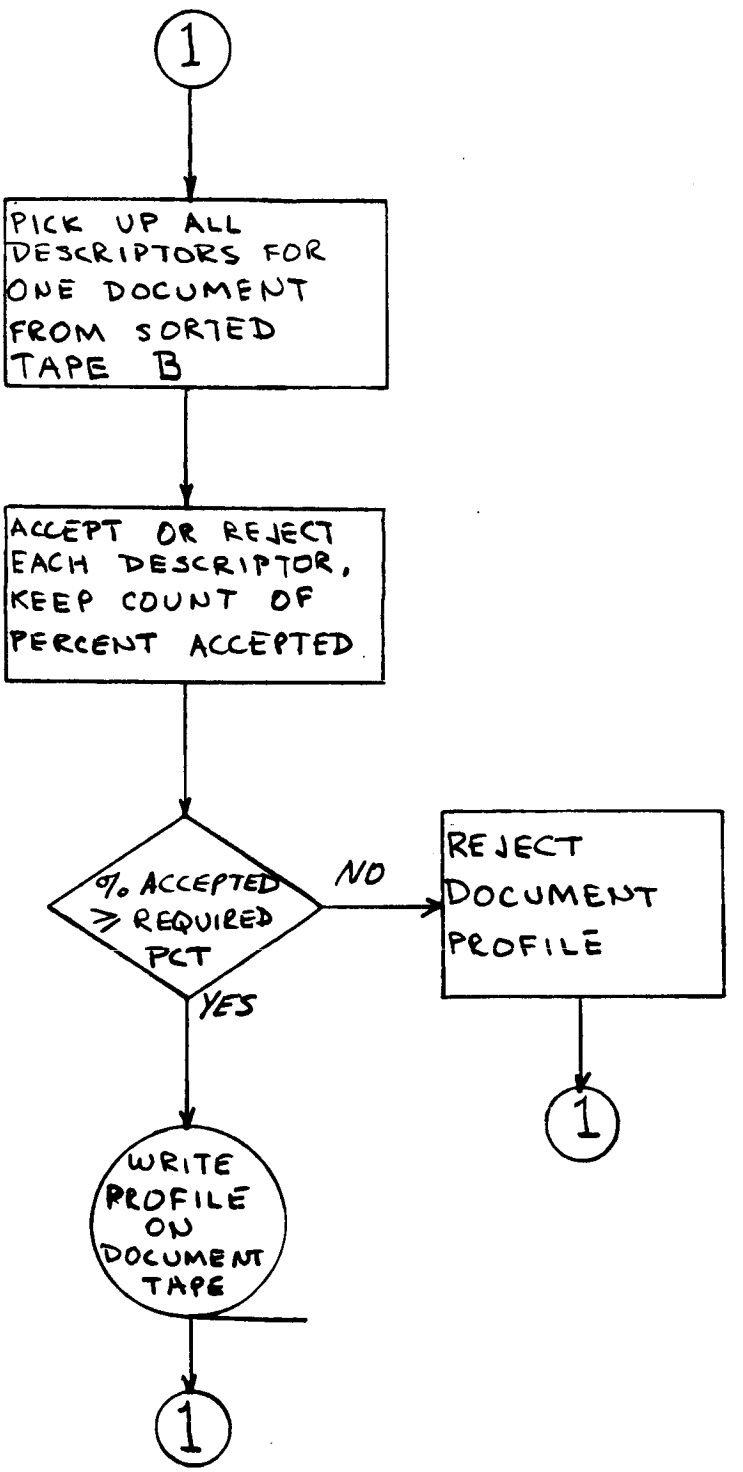
Subroutine PDTB

PDTB builds the binary document profile tape. The input tape is sorted tape B. The first descriptor for each document is its header. Following are its descriptors. Each descriptor is checked to see if it should be added to the document profile. When all of one document's descriptors have been processed, the percentage that have been accepted is compared against the control card parameter MIND. If the percent accepted is greater than or equal to MIND, the document is added to the binary tape; otherwise, it is rejected. The acceptance or rejection of every descriptor and document is noted on the system output tape.

CR-62021

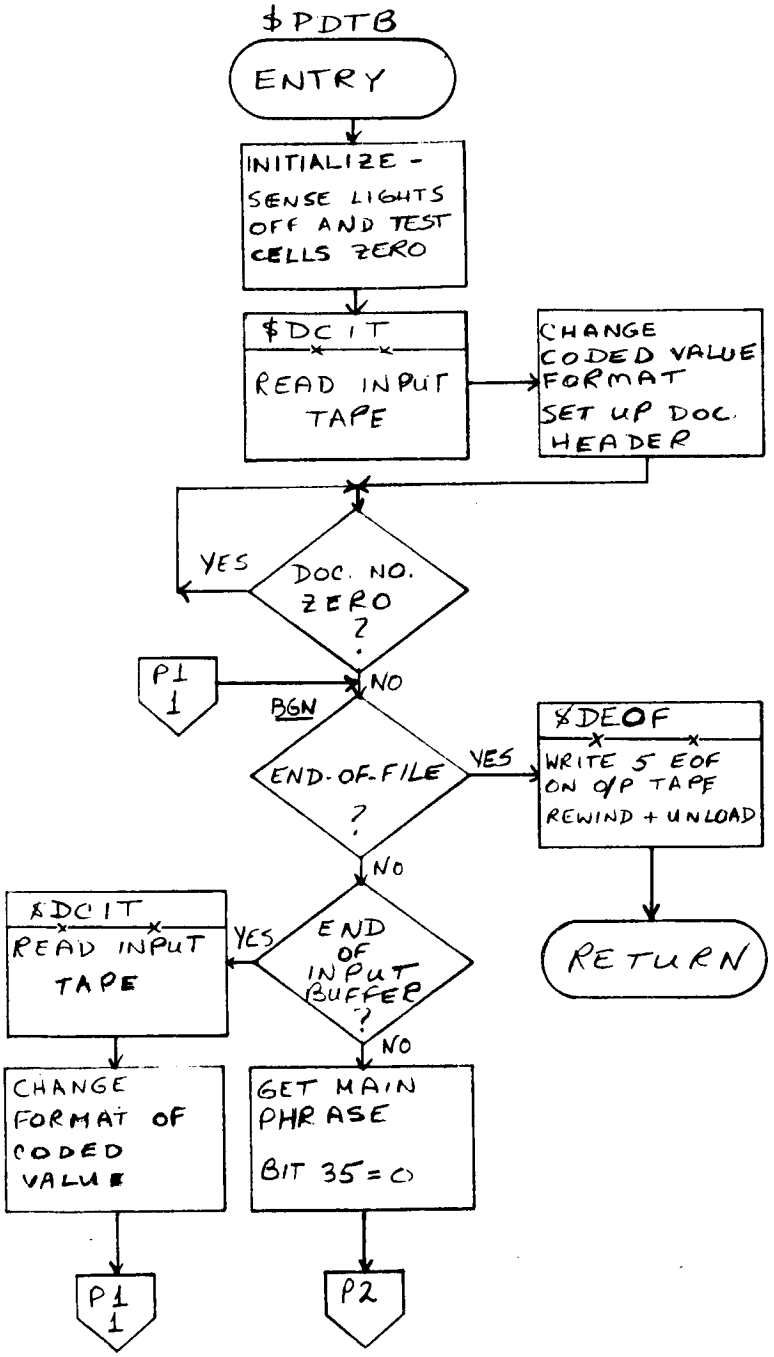
$PDTB

Block Diagram

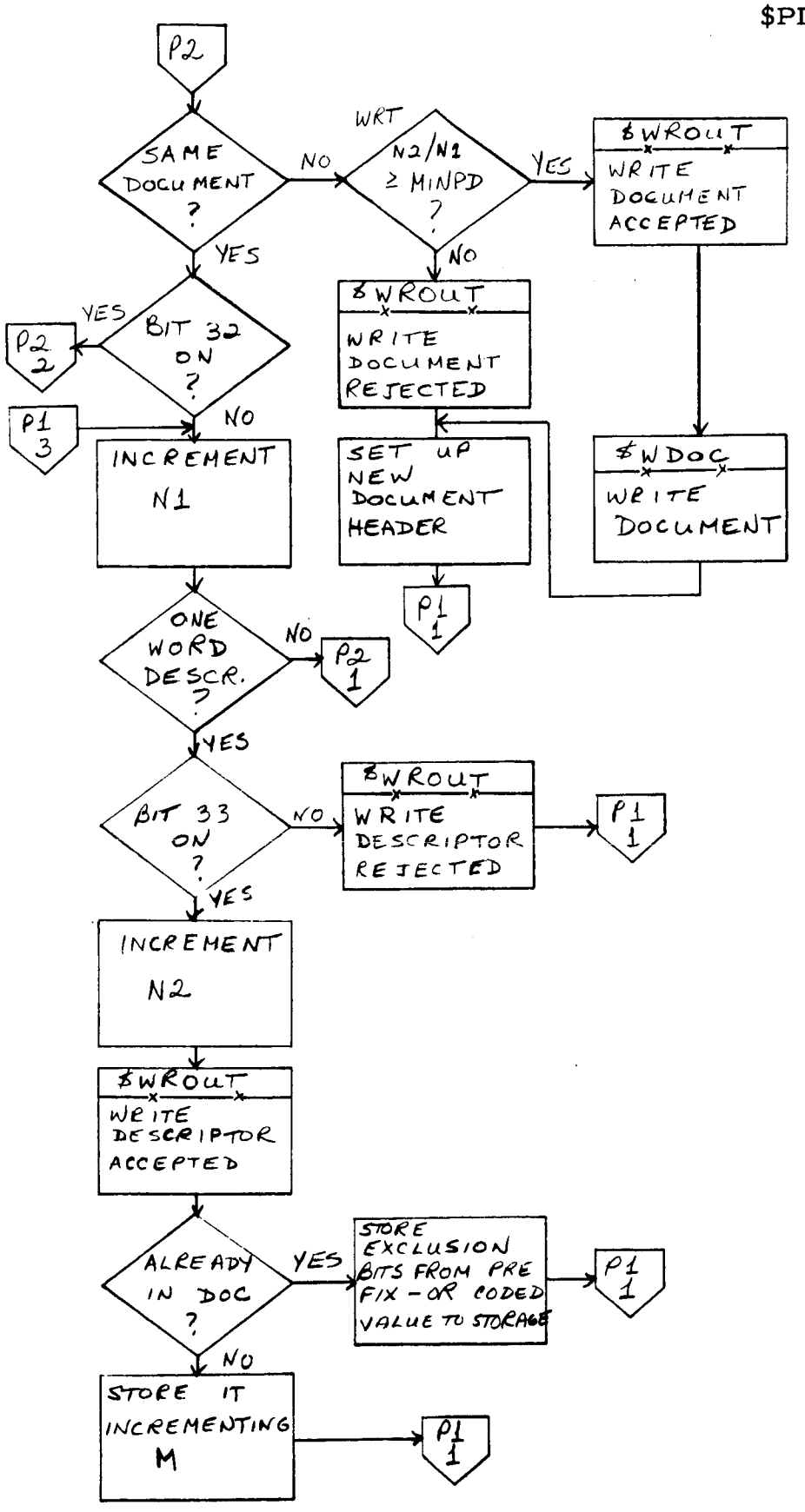


EVERY DESCRIPTOR, WHETHER ACCEPTED OR REJECTED IS SO NOTED ON THE BCD OUTPUT TAPE. THE SAME IS DONE FOR EACH PROFILE

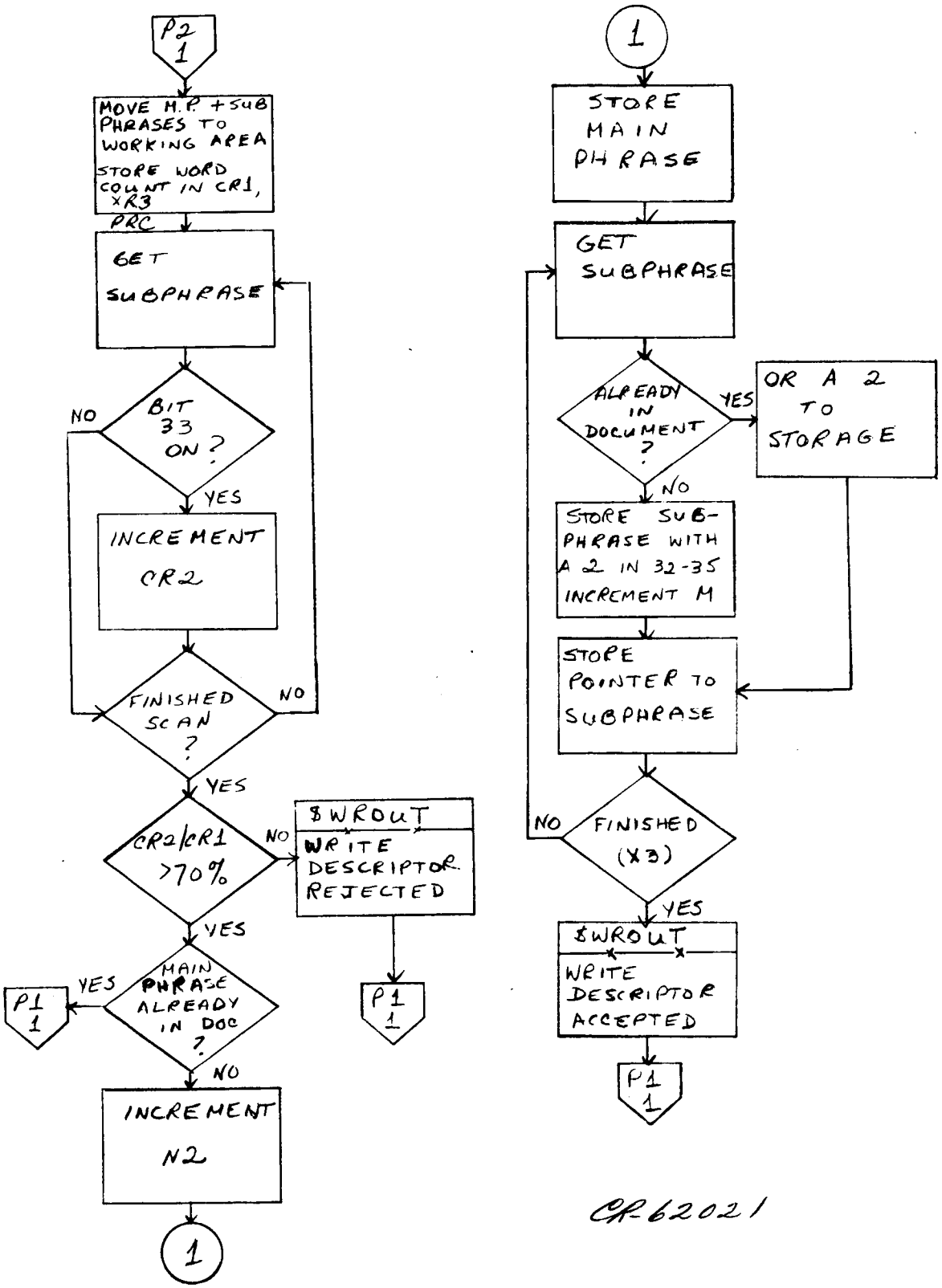
CR-62021


$PDTB
ENTRY
INITIALIZE - SENSE LIGHTS OFF AND TEST CELLS ZERO
$DC IT
READ INPUT TAPE
CHANGE CODED VALUE FORMAT SET UP DOC. HEADER
YES
DOC. NO. ZERO ?
P1 1
NO
BGN
END-OF-FILE ?
YES
$DEOF
WRITE 5 EOF ON O/P TAPE REWIND + UNLOAD
RETURN
No
$DC IT
READ INPUT TAPE
YES
END OF INPUT BUFFER ?
NO
CHANGE FORMAT OF CODED VALUE
GET MAIN PHRASE
BIT 35 = 0
P1 1
P2

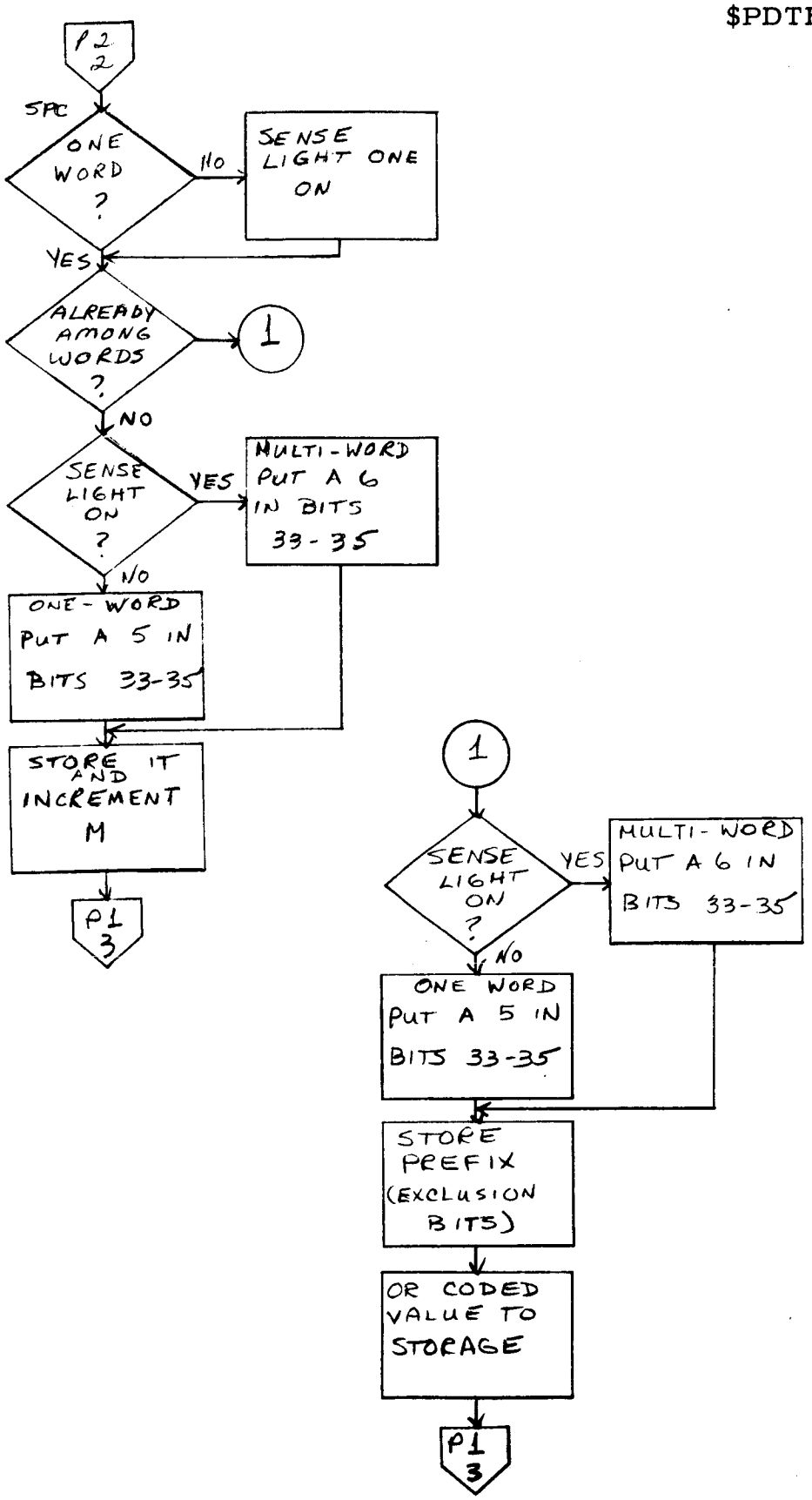
CF-62021


P2
SAME DOCUMENT ?
NO
WRT
N2/N1 ≥ MINPD ?
YES
$WROUT
WRITE DOCUMENT ACCEPTED
YES
NO
$WROUT
WRITE DOCUMENT REJECTED
YES
P2 2
BIT 32 ON ?
P1 3
NO
INCREMENT N1
SET UP NEW DOCUMENT HEADER
$WDOC
WRITE DOCUMENT
P1 1
ONE WORD DESCR. ?
NO
P2 1
YES
BIT 33 ON ?
NO
$WROUT
WRITE DESCRIPTOR REJECTED
P1 1
YES
INCREMENT N2
$WROUT
WRITE DESCRIPTOR ACCEPTED
ALREADY IN DOC ?
YES
STORE EXCLUSION BITS FROM PRE FIX - OR CODED VALUE TO STORAGE
P1 1
NO
STORE IT INCREMENTING M
P1 1

CR-62021


P2
1
MOVE M.P. + SUB PHRASES TO WORKING AREA
STORE WORD COUNT IN CR1, XR3
PRC
GET SUBPHRASE
BIT 33 ON ?
NO
YES
INCREMENT CR2
FINISHED SCAN ?
NO
YES
CR2/CR1 >70%
NO
$WROUT
WRITE DESCRIPTOR REJECTED
P1
1
YES
MAIN PHRASE ALREADY IN DOC ?
YES
P1
1
NO
INCREMENT N2
1
1
STORE MAIN PHRASE
GET SUBPHRASE
ALREADY IN DOCUMENT ?
YES
OR A 2 TO STORAGE
NO
STORE SUB-PHRASE WITH A 2 IN 32-35 INCREMENT M
STORE POINTER TO SUBPHRASE
FINISHED (X3)
NO
YES
$WROUT
WRITE DESCRIPTOR ACCEPTED
P1
1

CR62021

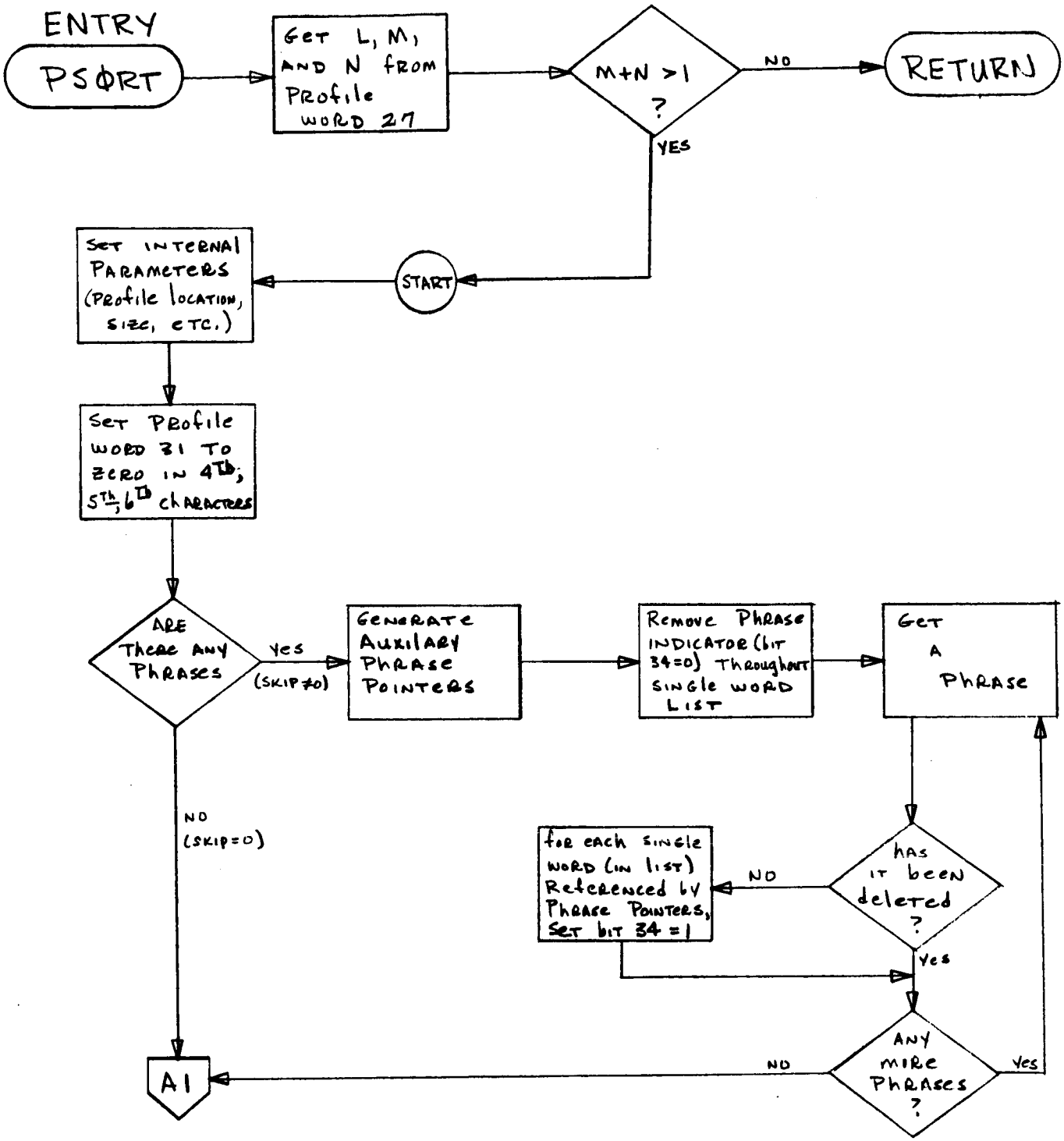

P2
2
SPC
ONE WORD ?
No
SENSE LIGHT ONE ON
YES
ALREADY AMONG WORDS ?
1
NO
SENSE LIGHT ON ?
YES
MULTI-WORD PUT A 6 IN BITS 33-35
NO
ONE-WORD PUT A 5 IN BITS 33-35
STORE AND INCREMENT M
P1
3
1
SENSE LIGHT ON ?
YES
MULTI-WORD PUT A 6 IN BITS 33-35
NO
ONE WORD PUT A 5 IN BITS 33-35
STORE PREFIX (EXCLUSION BITS)
OR CODED VALUE TO STORAGE
P1
3

CP-62021

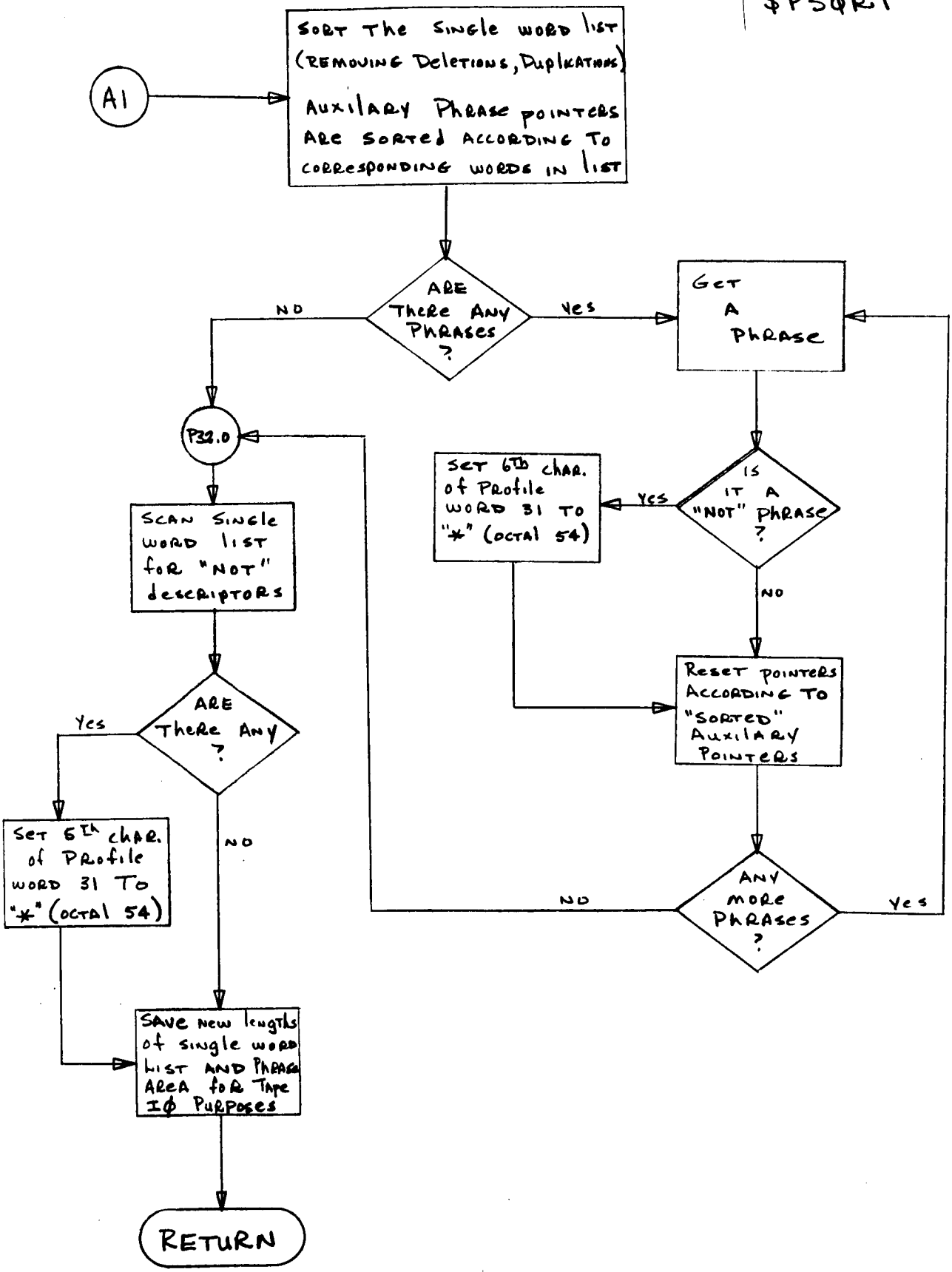
Subroutine PSORT

PSORT checks coded profiles (document or user) submitted to it to ascertain that the single-word list is sorted and that phrase pointers are set accordingly. If not, PSORT sorts and sets them.

CR-62001


ENTRY
PS$RT
Get L, M, and N from Profile word 27
M+N >1 ?
NO
RETURN
YES
START
Set internal parameters (Profile location, size, etc.)
Set Profile word 31 to zero in 4th, 5th, 6th characters
Are There any Phrases
Yes (SKIP≠0)
Generate Auxilary Phrase Pointers
Remove Phrase indicator (bit 34=0) throughout single word List
Get a Phrase
NO (SKIP=0)
for each single word (in list) Referenced by Phrase Pointers, Set bit 34 = 1
has it been deleted ?
NO
Yes
Any more Phrases ?
NO
Yes
A1

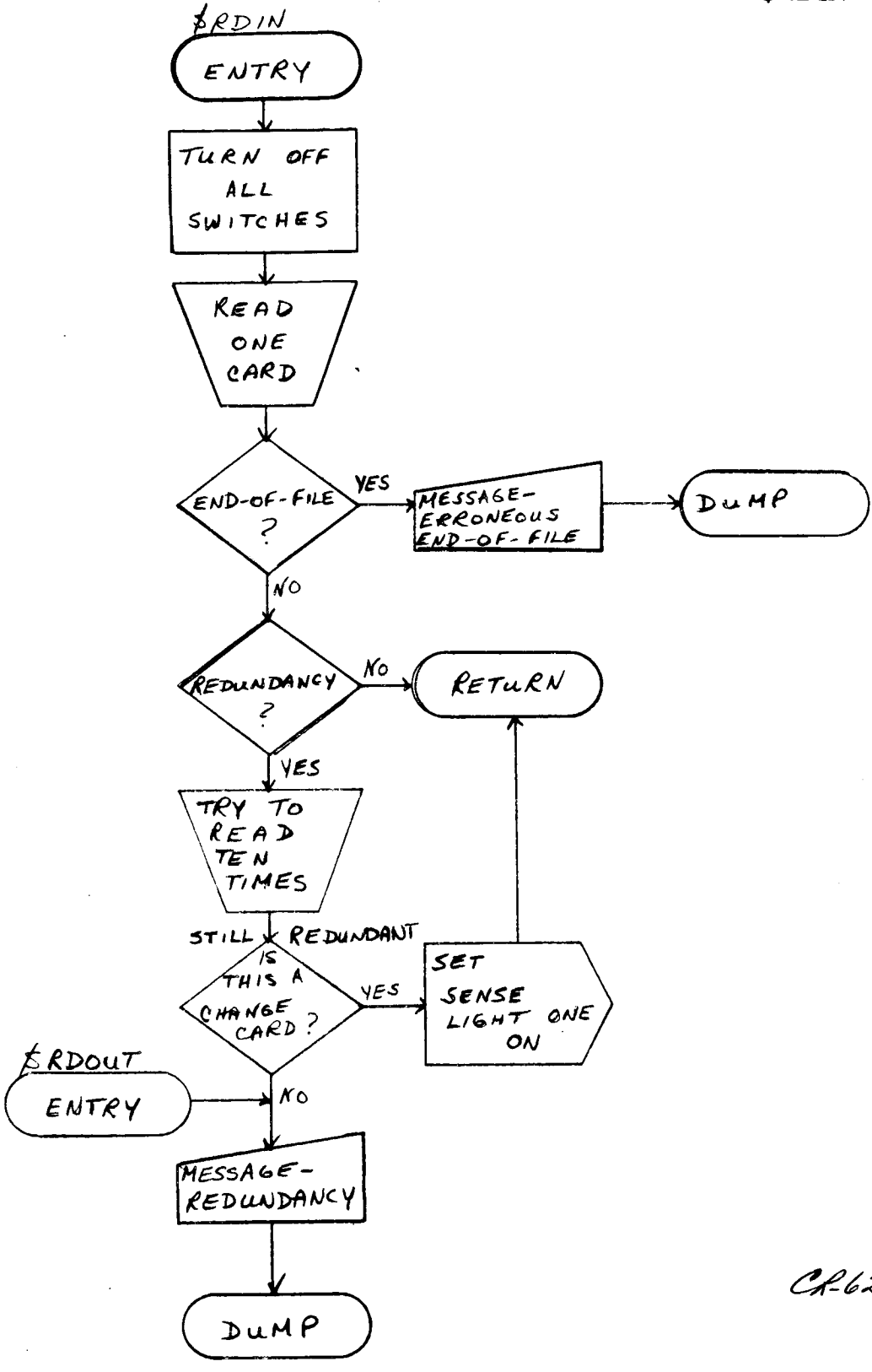
CR-62021


A1
SORT THE SINGLE WORD LIST (REMOVING DELETIONS, DUPLICATIONS)
AUXILARY PHRASE POINTERS ARE SORTED ACCORDING TO CORRESPONDING WORDS IN LIST
ARE THERE ANY PHRASES ?
NO
YES
GET A PHRASE
P32.0
SCAN SINGLE WORD LIST FOR "NOT" DESCRIPTORS
IS IT A "NOT" PHRASE ?
YES
SET 6TH CHAR. of PROFILE WORD 31 TO "*" (OCTAL 54)
NO
RESET POINTERS ACCORDING TO "SORTED" AUXILARY POINTERS
ARE THERE ANY ?
YES
SET 5TH CHAR. of PROFILE WORD 31 TO "*" (OCTAL 54)
NO
ANY MORE PHRASES ?
NO
YES
SAVE NEW LENGTHS of SINGLE WORD LIST AND PHRASE AREA FOR TAPE I/O PURPOSES
RETURN

Subroutine RDIN

This subroutine, used by MAIN and GTUSE, reads the system input tape. On any redundancy, an attempt is made to read the record ten times. If the redundant condition persists for a control card, a message indicating the redundancy is printed and a core dump is taken. RDIN also checks for EOF and prints a message if the condition exists and dumps core.

CR-62021

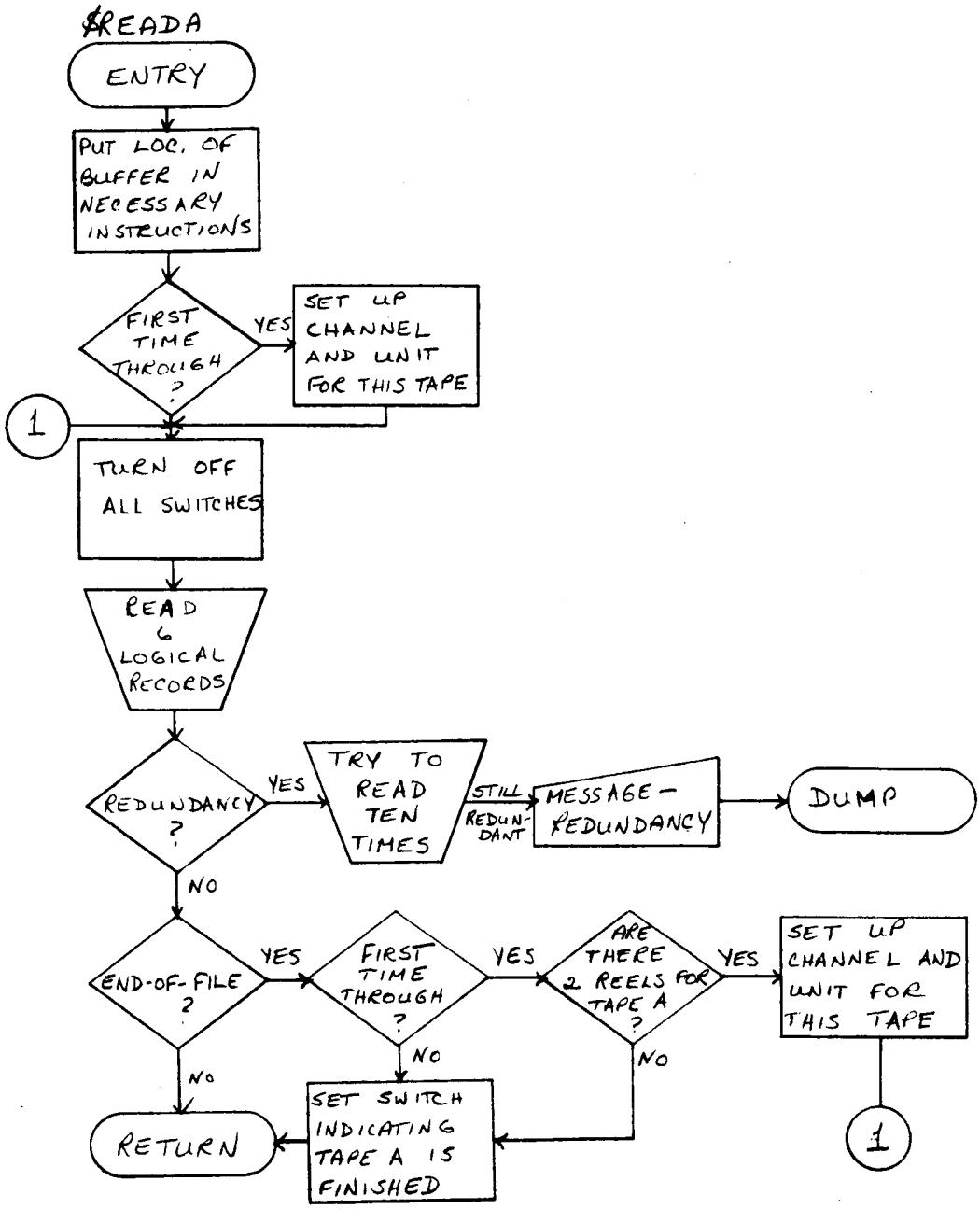

$RDIN
ENTRY
TURN OFF ALL SWITCHES
READ ONE CARD
END-OF-FILE ?
YES
MESSAGE-ERRONEOUS END-OF-FILE
DUMP
NO
REDUNDANCY ?
NO
RETURN
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
IS THIS A CHANGE CARD ?
YES
SET SENSE LIGHT ONE ON
$RDOUT
ENTRY
NO
MESSAGE-REDUNDANCY
DUMP

CR-62021

Subroutine READA

This subroutine reads a physical record consisting of six logical records from tape A (sorted input tape). On a redundancy condition, an attempt is made to read a record ten times before a message is given indicating the redundancy and a dump of core is taken. The routine allows tape A to have two reels.

CR-62021

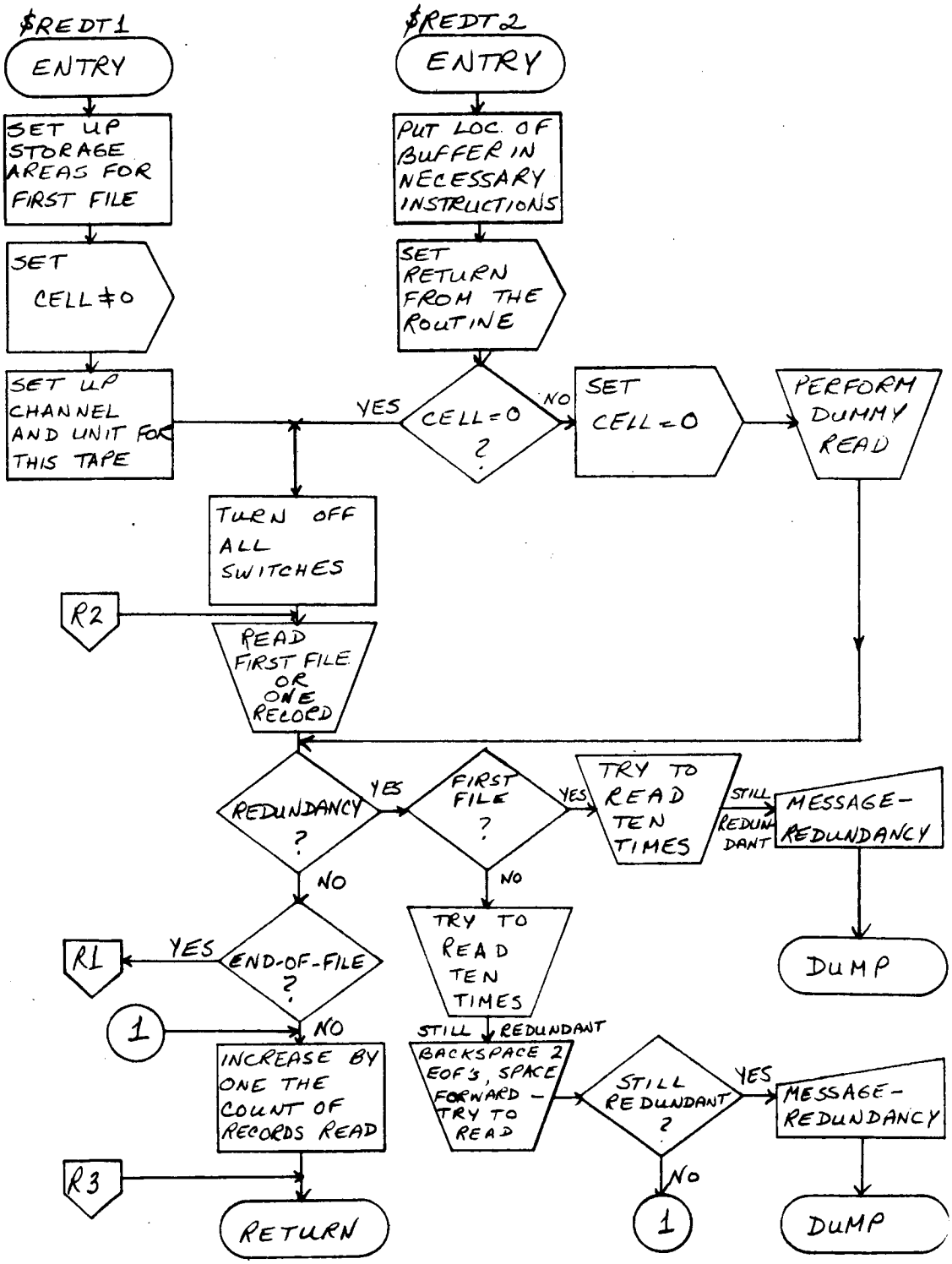

$READA
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
1
TURN OFF ALL SWITCHES
READ 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
MESSAGE - REDUNDANCY
DUMP
NO
END-OF-FILE ?
YES
FIRST TIME THROUGH ?
YES
ARE THERE 2 REELS FOR TAPE A ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
1
NO
NO
NO
SET SWITCH INDICATING TAPE A IS FINISHED
RETURN

CR-62021

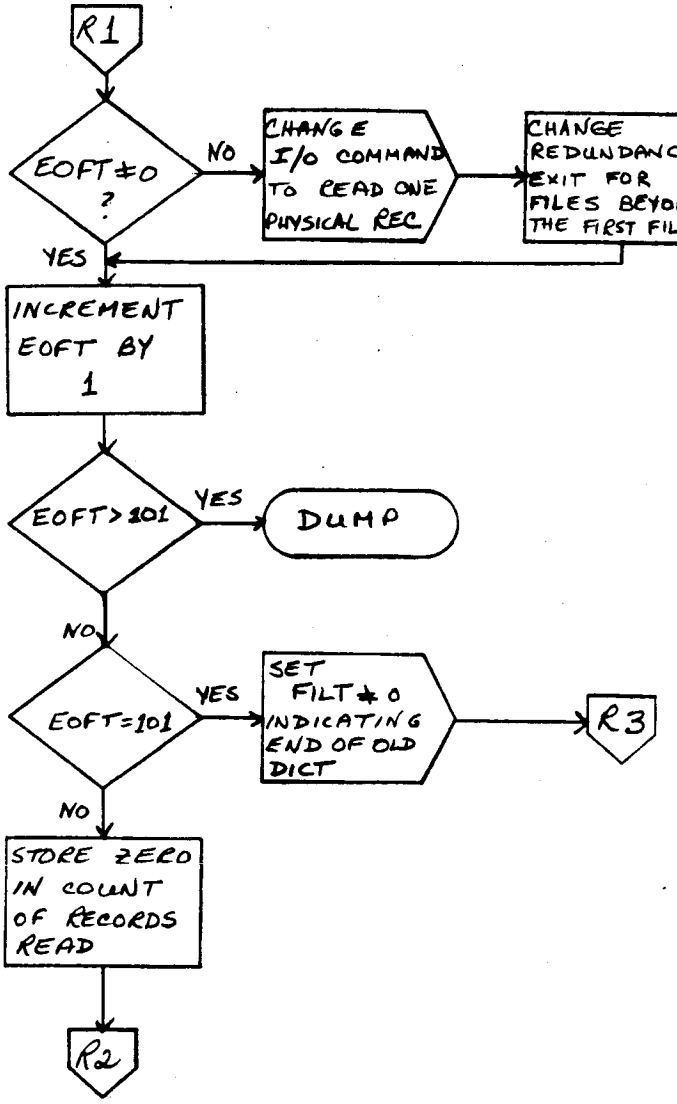
Subroutine REDT

This subroutine reads the old vocabulary. It has two entry points, REDT1 to read the first file and REDT2 to read the other one hundred files. A redundancy condition on the first file causes an attempt to be made to read the file ten times. If this fails, a message is printed and a dump taken. A redundant condition on any other file causes the tape to be reread ten times if necessary. If this fails, the tape is backspaced over two EOF's, spaced forward again, and another attempt is made to read. If the redundancy still exists a message is given and a dump taken.

CR-62021


$REDT1
ENTRY
SET UP STORAGE AREAS FOR FIRST FILE
SET CELL≠0
SET UP CHANNEL AND UNIT FOR THIS TAPE
$REDT2
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
SET RETURN FROM THE ROUTINE
CELL=0 ?
YES
NO
SET CELL=0
PERFORM DUMMY READ
TURN OFF ALL SWITCHES
R2
READ FIRST FILE OR ONE RECORD
REDUNDANCY ?
YES
FIRST FILE ?
YES
TRY TO READ TEN TIMES
STILL REDUNDANT
MESSAGE-REDUNDANCY
DUMP
NO
NO
R1
YES
END-OF-FILE ?
1
NO
INCREASE BY ONE THE COUNT OF RECORDS READ
R3
RETURN
TRY TO READ TEN TIMES
STILL REDUNDANT
BACKSPACE 2 EOF's, SPACE FORWARD - TRY TO READ
STILL REDUNDANT ?
YES
MESSAGE-REDUNDANCY
DUMP
No
1

CR-62021


R1
EOFT≠0 ?
NO
CHANGE I/O COMMAND TO READ ONE PHYSICAL REC
CHANGE REDUNDANCY EXIT FOR FILES BEYOND THE FIRST FILE
YES
INCREMENT EOFT BY 1
EOFT>101
YES
DUMP
NO
EOFT=101
YES
SET FILT ≠ 0 INDICATING END OF OLD DICT
R3
NO
STORE ZERO IN COUNT OF RECORDS READ
R2

CR-62021

Subroutine SORT

General Specifications

The SORT subroutine is a generalized internal sort and tape merge program that sorts fixed-length blocked or unblocked binary records. The SORT routine is entered in Phases 3, 5, and 6 of the program. In Phase 3, it sorts user and document profiles and vocabulary changes into coded descriptor order. In Phases 5 and 6, it sorts edited document profiles and edited user profiles, respectively, into profile and serial number order.

The sort key must be located at the beginning of each logical record and must be a multiple of six bits in length. Input and output files may each use one alternate unit. All input tapes must have a BCD trailer label. For multi-reel files, each tape must end with an end-of-file and a 120-character 1EOR record. The last reel, or the single reel of a one-reel file should have an end-of-file and a 120-character 1EOT record. For the merge pass, the package requires four intermediate tapes, two on channel A and two on channel B, the blocking of these tapes may be different from the initial or final blocking.

The SORT package consists of three major subroutines, EDIT, SORT, and WRITE. It requires the FORTRAN II Input/Output package IOP for its I/O functions, and the subroutine DMP for catastrophic errors. The SORT package must be assembled by the user, who provides variables in cards 60-400 as required (see listing). It uses approximately 1200)10 locations plus BSS areas, the size of which depends on the physical size of the records to be sorted. All core not needed by the object program should be allocated to SORT for internal sorting. The calling sequence to SORT is TSX SORT,4 with a return of 1,4.

Subroutine EDIT

This subroutine has two entries, EDITA and EDIT, which are the first and second executable instructions. EDITA is the entry for the first call, to accomplish any initialization necessary in view of the fact that the sort may be restarted due to tape error. The routine places in the AC decrement (bits 0-18) the number of words in the record to be sorted, and in the MQ address (19-36) the number of characters in the sort key. It then returns to 1,4. EDIT is the normal entry. The calling sequence is TSX EDIT,4, end-of-file return, normal return.

Each time it is called, the subroutine will store a record starting at 1,2. It reads as much or as little data from the mediary tape as it needs to. The record placed at 1,2 is arranged with the sort key at the beginning, since the sort program will sort on only one field. Conversions to modify the natural sequence or complementing of fields to achieve a descending sort may be done prior to storing the record at 1,2. If there is no more data, return is to 1,4. The normal return is to 2,4. The subroutine also handles any errors encountered as described below.

CR-62021

Subroutine SORT

A replacement chain sort is used for the internal (string-forming) phase of the sort program. The chain is formed as follows: With each record of data is associated one word of control information. The decrement of this word contains the location of the next smaller record currently in core. The address portion contains the location of the next greater record (actually the two's complement of the address of the word preceding the first one of the record in question). The decrement of the control word of the smallest record in core (the beginning of the chain) contains zero, as does the address of the control word of the largest record (the end of the chain). Thus, given the location of the starting record of the chain, one can reference each record in the desired logical sequence, and the data is effectively sorted. Note that the physical arrangement of the data in memory is irrelevant to the structure of the chain. Note also that the insertion of a new record into the chain requires the creation of a control word for the new record and the modification of the control words of the two records which will immediately precede and follow it in the chain. No physical movement or shifting of the data is necessary.

The sort program starts by storing records sequentially in memory, preceding each one by its control word. Each record received from EDIT is immediately put in the chain by the subroutine INSERT. Since some pre-sequencing is expected to exist in the data, the INSERT subroutine starts by comparing the new record with the one most recently inserted. Depending on the result of this comparison, the subroutine searches the chain in either the forward (increasing) or backward (decreasing) direction until the proper place for the new record is found. In the case of equal records, the order of input is preserved. The new record is then inserted in the chain, as described above. A record is kept of the location of the starting member of the chain, and updated when necessary.

If all the input has been read in before available core space is filled, the replacement phase and tape merge, described below, are skipped, and the routine goes directly into the WRITE subroutine. Otherwise, replacement and tape merge are necessary.

The replacement phase of the internal sort is started when there is room for only one more record in memory. Starting with the first merge tape and the first record in the chain, the procedure is as follows:

(1) Write the first record in the chain on the merge tape;
(2) Read a new (replacement) record into the one vacant space in core;
(3) Insert the new record in the chain (if it is greater than or equal to the previously written record, it will be in the same string. If less, it will be in the next string);
(4) Remove the previously written record from the chain, vacating its space;
(5) Repeat from step (1), writing out the follower in the chain of the previously written record.

This procedure continues until either the end of the input is reached or the last member of the chain has been written out. In the latter case, the chain is now composed of those replacement records which were less than the record just written when they were read in (otherwise, they would have been written out in that string and themselves replaced). An end-of-file is written after the completed string, and a new string is started on the next merge tape, again beginning with the first member of the chain. When there is no more input, the data is written as above without being replaced. The current string is finished and, unless the input end-of-file had coincided with the start of a new string, one final string is written.

Thus, throughout most of the internal sort phase, the operations of reading, writing and sorting are overlapped and all take place in the same storage area. The lengths of the strings produced depend on the pre-sequencing of the input. In the worst case (perfect inverse order) the strings contain as many records as can be held in memory at one time. In any other case, the strings will be considerably longer.

The technique used is that of a simple balanced merge. The final pass, in which some number of strings $\leq$ the merge order (one string to a tape) are merged into one output string, gives the sorted data directly to the WRITE subroutine.

Subroutine WRITE

This subroutine has one entry, WRITE, which is the first executable instruction. Each time it is called, it finds one sorted record of data starting at 1,2, in ascending logical order. The routine may re-convert or complement the sort key, as necessary, and then will write the appropriate record(s) of the output report on the system's output tape. When all data has been processed, the sort program must make one last entry to WRITE with a word of all one's, $777777777777)_8$, at 1,2.

The FORTRAN II Input/Output Package (IOP) is called into core by using the * IOP card in the deck setup. The program uses IOP to take care of much of its input/output functions.

Error Messages

The sort program keeps two checks on its processing. First, the records input are counted, and this number compared with the number processed by each merge pass and the number eventually output. This is a check for records lost or duplicated due to program or tape error. The program also checks that records written out, either on merge tapes or given to WRITE are, in fact, in the proper sequence. Violations of the sort order could be due to program error or undetected tape error.

CR 62021

UNEXPECTED EOF ON MEDIARY TAPE.
The EDIT subroutine took the EOF return the first time it was entered. There may have been no data at all on input tape or none valid.

ERROR READING MERGE TAPE XX
This message is written after an error return from a BBREAD of a merge (scratch) tape. If it is not the last pass of the sort and it is the first time this has occurred, the program prints on-line: ERROR READING TAPE XX. MOUNT NEW TAPE AND PRESS START TO CONTINUE JOB. The program then stops at HPR 0. The operators are to change tapes and let the program proceed, in which case it prints "RESTARTING SORT" off-line and re-trys the sort from the beginning. If, however, this re-occurs or occurs the first time during the last pass, the program will follow the first message with "END PROCESSING" and will transfer $DMP.

UNEXPECTED EOF ON MERGE TAPE XX.
END PROCESSING.
This unlikely message indicates that the sort program's count of the number of strings on the merge tape was incorrect. This could be due to program error, machine error or tape error. The program transfers to $DMP.

PH.X RECORD COUNTS DISAGREE. / END PROCESSING.
PH.X DATA OUT OF SORT. / END PROCESSING.
Phase 1 (PH.X) is the internal sort and string-forming section (the first pass over the data); Phase 2 is the tape merge. Phase 3 is the final merge pass and writing of the report itself. The errors are most likely due to tape failure, although they could be caused by improper tampering with data and storage by the EDIT and WRITE subroutines. The program transfers to $DMP.

Sort Tape Formats

Tape A - All Input Data

Tape A is sorted only on word 1, that is, according to the coded value of the descriptor. Hence all header records will appear before descriptor records on the sorted Tape A.

Descriptor Records:

Word	
1	Coded value of descriptor
2	Same as the third word put out by CODER. (Bits S1-17 have type, col. 71 of input card, etc. Bits 18-35 have serial number.)
3	Zero if dictionary change, otherwise the document or user profile number.
4-23	Twenty word alphanumeric descriptor. (If this descriptor is a subphrase, these twenty words will be either blanks or zeros depending on the output from CODER.)

CR-62021

24	Coded value if this is a dictionary secondary descriptor, otherwise blanks or zeros.
25-27	All blanks or zeros.

Header Records:

Word	
1	All zeros
2	Same as for a descriptor record
3	Document or user profile number
4-27	Twenty-four word heading.

Tapes B and C

Tape B has document records and Tape C has user records. Both documents and users have the same basic format. Tapes B and C are sorted on profile number and serial number. Hence all descriptors having the same profile number will be together preceded by the header for that profile.

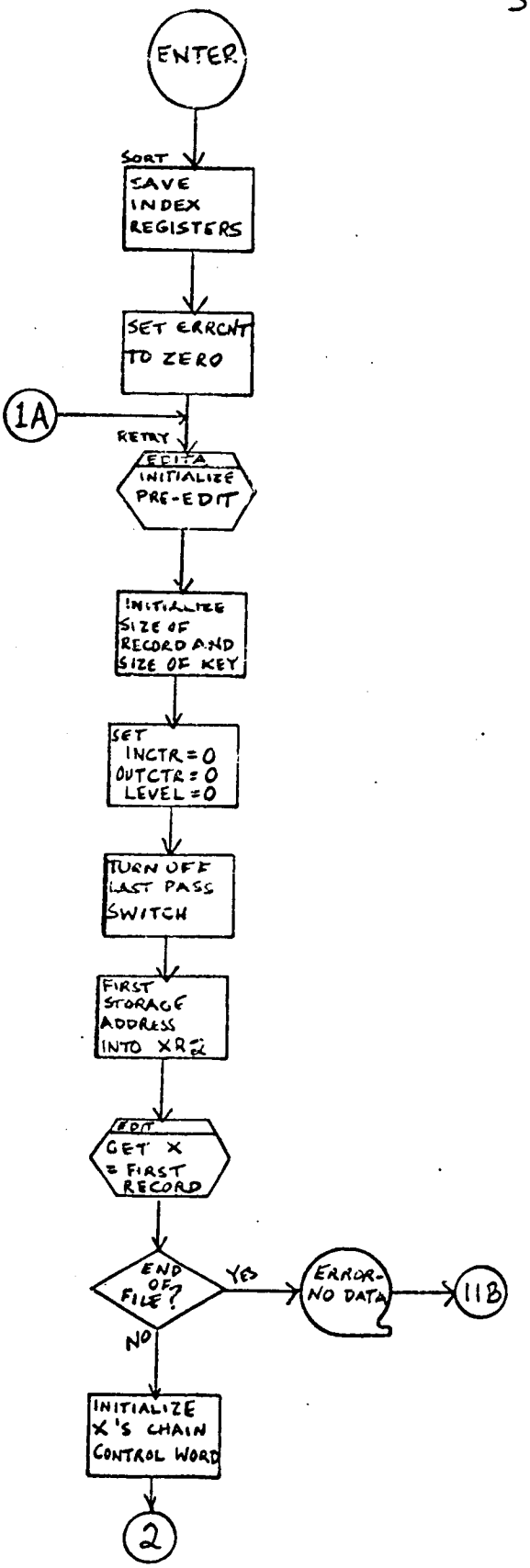
Descriptor Records:

Word	
1	Document or user profile number
2	Has the same content as word 3 of the CODER output but with the two halves of the word interchanged: bits S1-17, serial number, bits 18-35, various bits indicating type, col. 71, etc.
3	Coded value of descriptor
4-23	Twenty-word alphanumeric descriptor (or all blanks if a subphrase.)
24-27	All blanks.

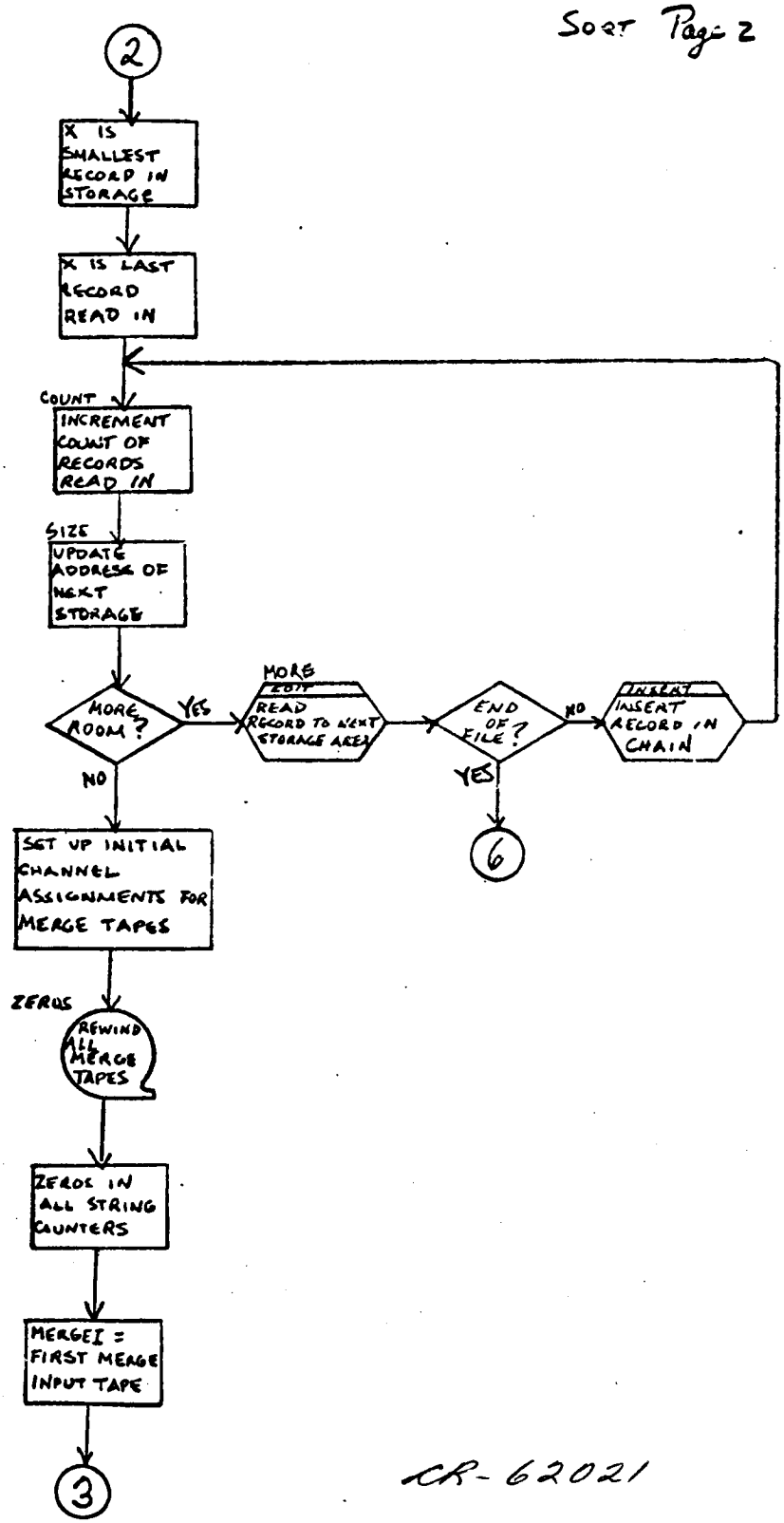
Header Records:

Word	
1-2	Same as descriptor records
3	All zeros
4-27	Twenty-four word heading.

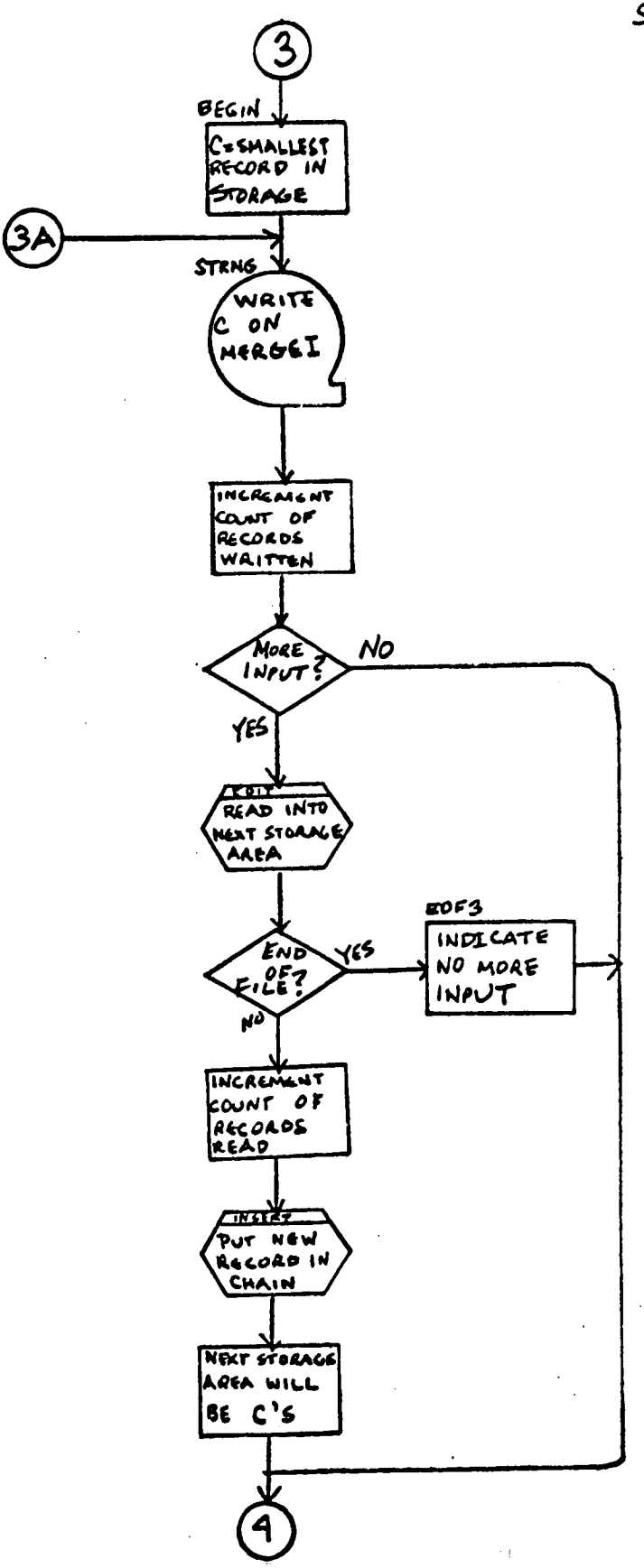
CR-62021

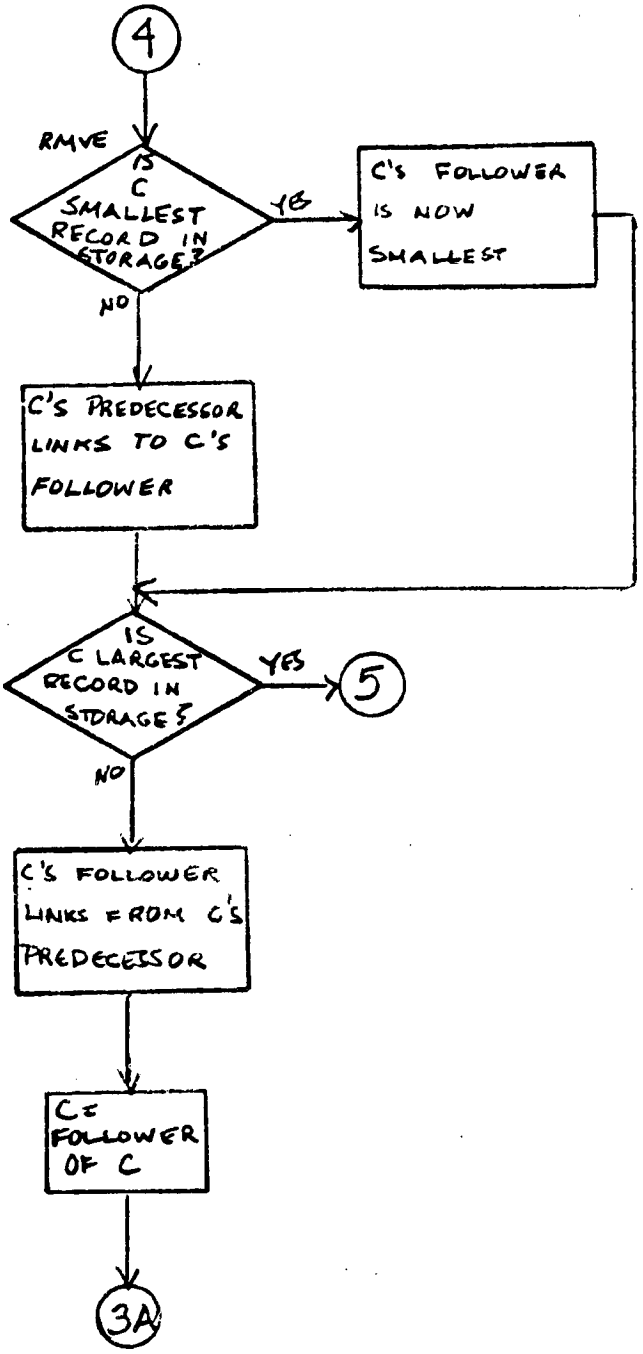

ENTER
SORT
SAVE INDEX REGISTERS
SET ERRCNT TO ZERO
1A
RETRY
EDITA
INITIALIZE PRE-EDIT
INITIALIZE SIZE OF RECORD AND SIZE OF KEY
SET INCTR=0 OUTCTR=0 LEVEL=0
TURN OFF LAST PASS SWITCH
FIRST STORAGE ADDRESS INTO XR2
EDIT
GET X = FIRST RECORD
END OF FILE?
YES
ERROR-NO DATA
11B
NO
INITIALIZE X'S CHAIN CONTROL WORD
2

CR-62021

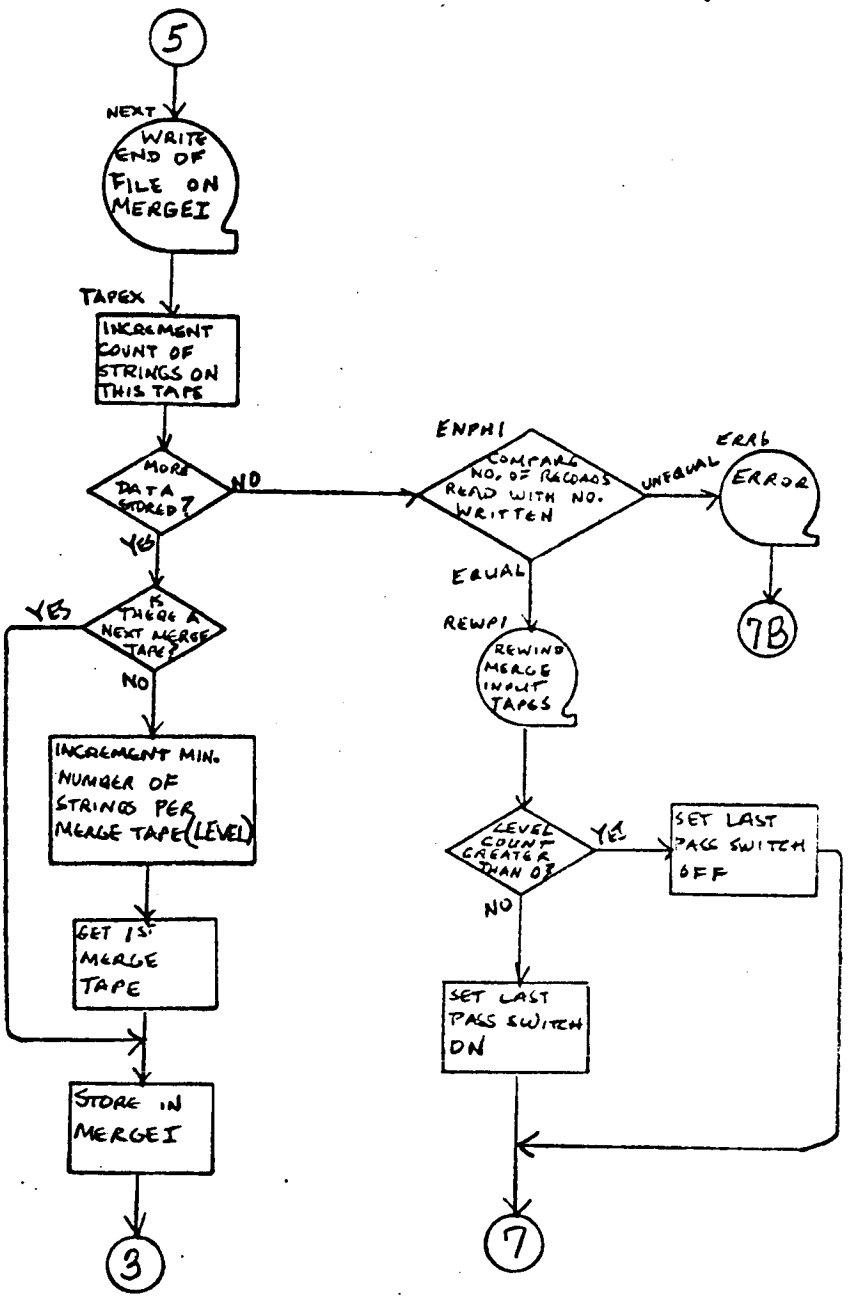

2
X IS SMALLEST RECORD IN STORAGE
X IS LAST RECORD READ IN
COUNT
INCREMENT COUNT OF RECORDS READ IN
SIZE
UPDATE ADDRESS OF NEXT STORAGE
MORE ROOM?
YES
MORE
READ RECORD TO NEXT STORAGE AREA
END OF FILE?
NO
INSERT
INSERT RECORD IN CHAIN
YES
6
NO
SET UP INITIAL CHANNEL ASSIGNMENTS FOR MERGE TAPES
ZEROS
REWIND ALL MERGE TAPES
ZEROS IN ALL STRING COUNTERS
MERGEI = FIRST MERGE INPUT TAPE
3

CR-62021

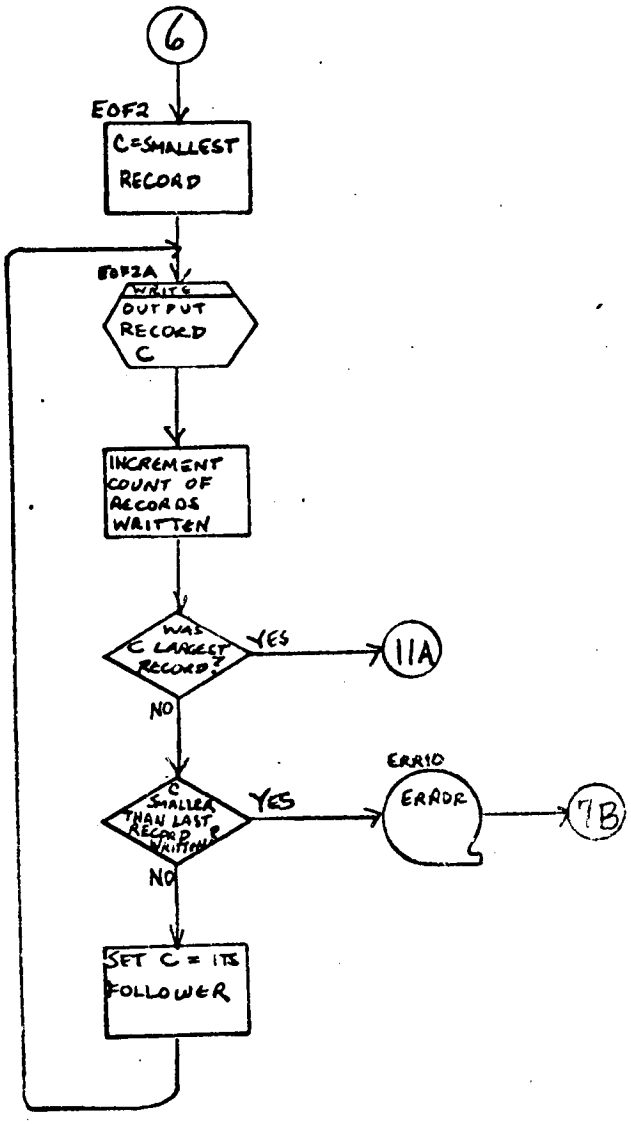

3
BEGIN
C=SMALLEST RECORD IN STORAGE
3A
STRNG
WRITE C ON MERGE1
INCREMENT COUNT OF RECORDS WRITTEN
MORE INPUT?
NO
YES
READ
READ INTO NEXT STORAGE AREA
EOF3
END OF FILE?
YES
INDICATE NO MORE INPUT
NO
INCREMENT COUNT OF RECORDS READ
INSERT
PUT NEW RECORD IN CHAIN
NEXT STORAGE AREA WILL BE C'S
4


4
RMVE
IS C SMALLEST RECORD IN STORAGE?
YES
C's FOLLOWER IS NOW SMALLEST
NO
C's PREDECESSOR LINKS TO C's FOLLOWER
IS C LARGEST RECORD IN STORAGE?
YES
5
NO
C's FOLLOWER LINKS FROM C's PREDECESSOR
C= FOLLOWER OF C
3A

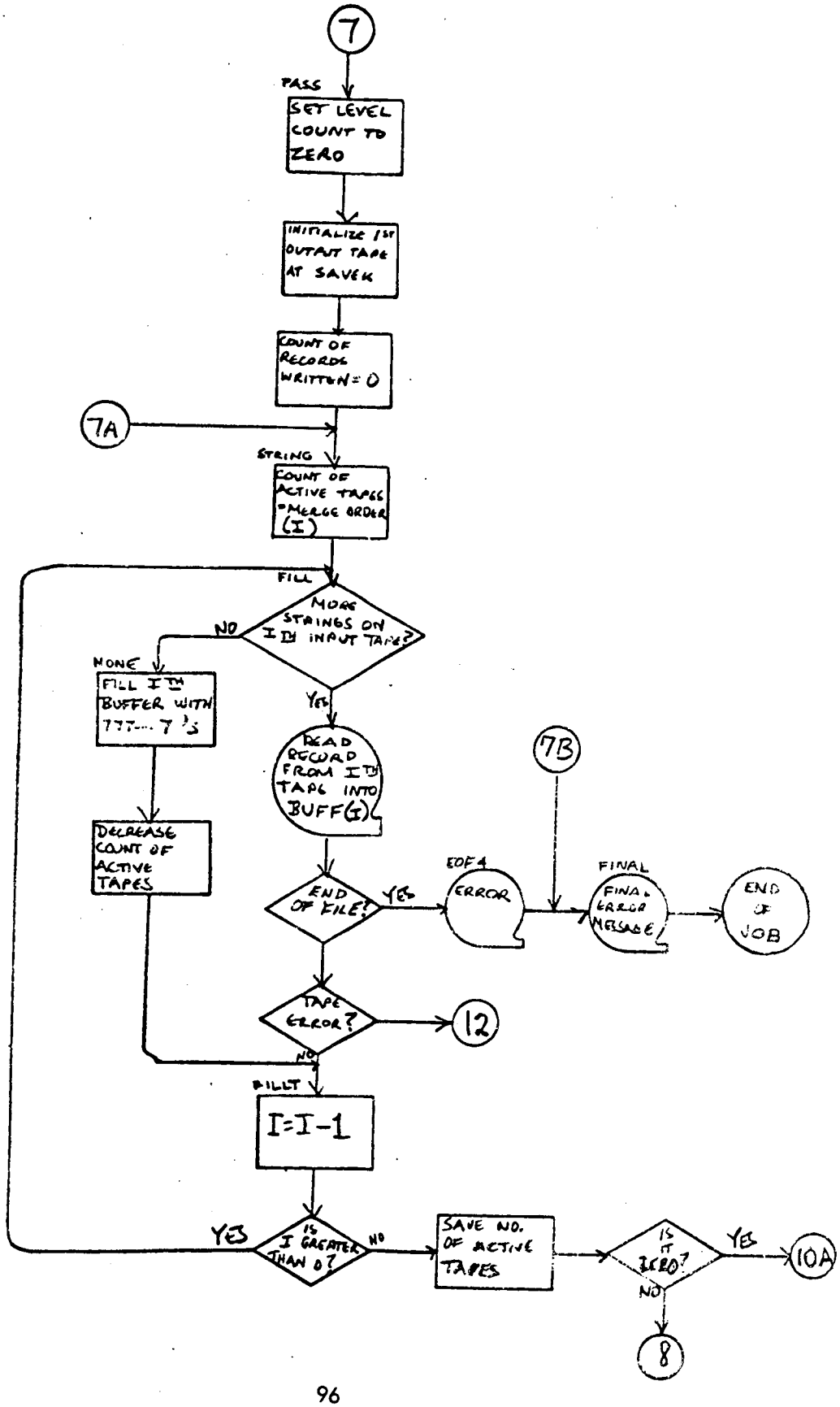
CR-62021


5
NEXT
WRITE END OF FILE ON MERGEI
TAPEX
INCREMENT COUNT OF STRINGS ON THIS TAPE
MORE DATA STORED?
NO
YES
ENPH1
COMPARE NO. OF RECORDS READ WITH NO. WRITTEN
UNEQUAL
ERR6
ERROR
7B
EQUAL
REWP1
REWIND MERGE INPUT TAPES
IS THERE A NEXT MERGE TAPE?
YES
NO
INCREMENT MIN. NUMBER OF STRINGS PER MERGE TAPE (LEVEL)
GET 1ST MERGE TAPE
STORE IN MERGEI
3
LEVEL COUNT GREATER THAN 01
YES
NO
SET LAST PASS SWITCH OFF
SET LAST PASS SWITCH ON
7

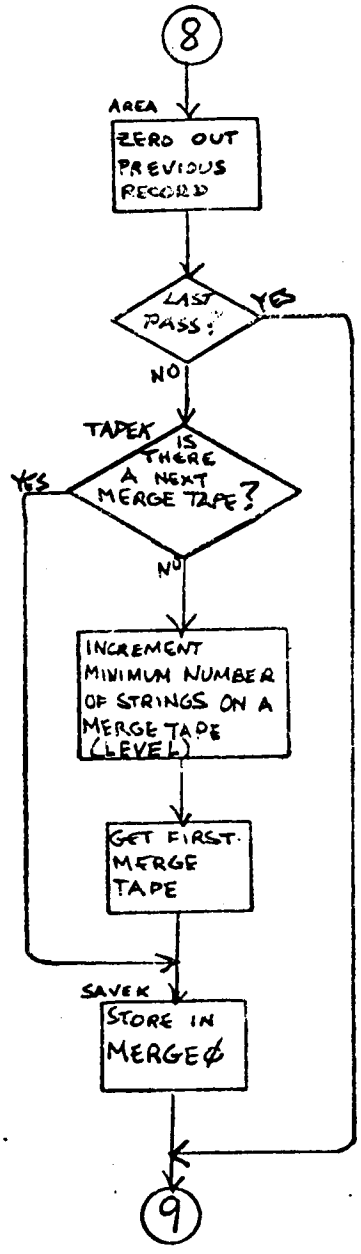
CR-62021


6
EOF2
C=SMALLEST RECORD
EOF2A
WRITE OUTPUT RECORD C
INCREMENT COUNT OF RECORDS WRITTEN
WAS C LARGEST RECORD?
YES
11A
NO
C SMALLER THAN LAST RECORD WRITTEN?
ERR10
YES
ERROR
7B
NO
SET C = ITS FOLLOWER

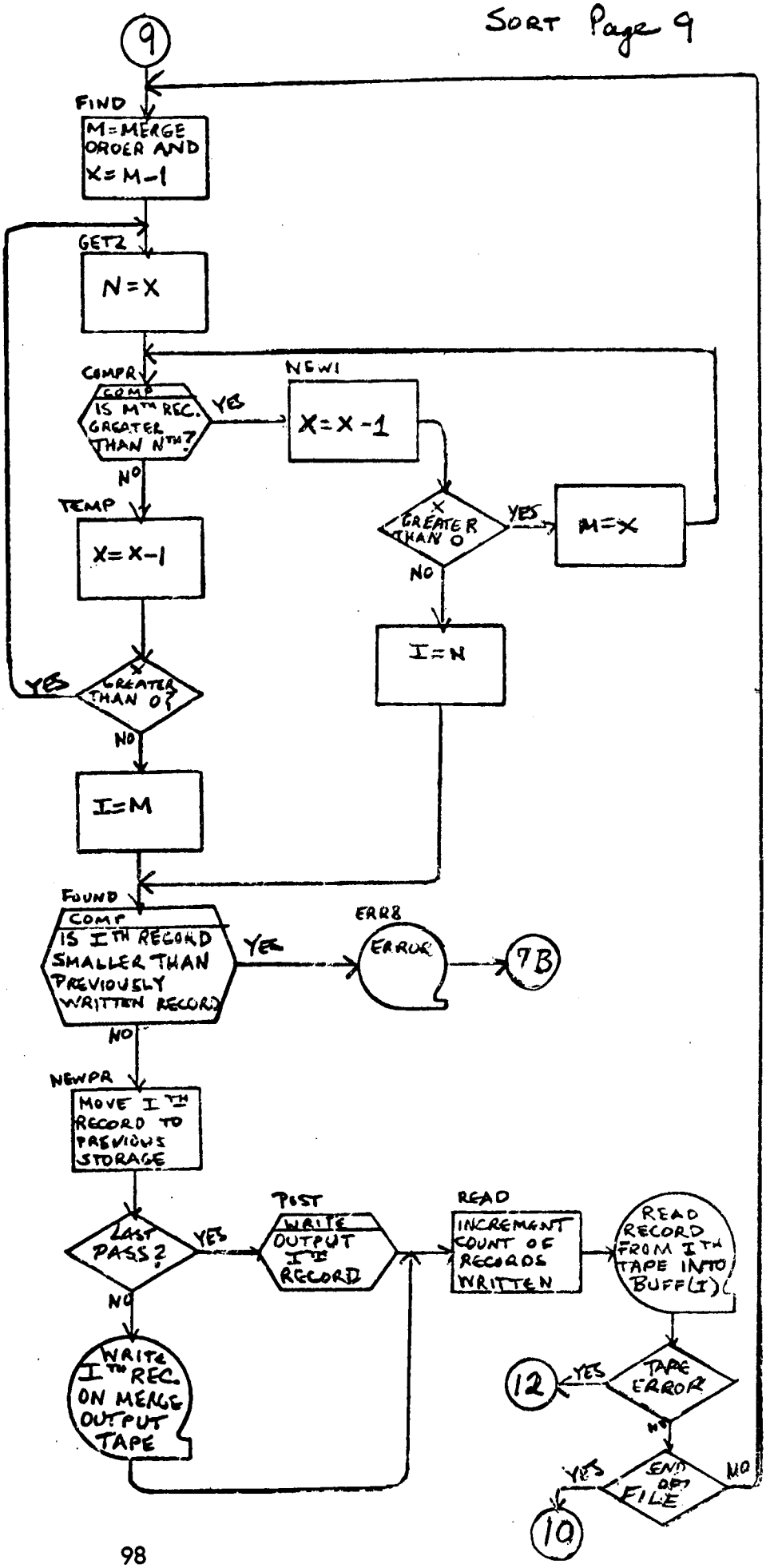
CR-62021


7
PASS
SET LEVEL COUNT TO ZERO
INITIALIZE 1ST OUTPUT TAPE AT SAVE4
COUNT OF RECORDS WRITTEN = 0
7A
STRING
COUNT OF ACTIVE TAPES = MERGE ORDER (I)
FILL
MORE STRINGS ON I TH INPUT TAPE?
NO
NONE
FILL I TH BUFFER WITH 777--- 7's
DECREASE COUNT OF ACTIVE TAPES
YES
READ RECORD FROM I TH TAPE INTO BUFF(I)
7B
END OF FILE?
YES
EOF4 ERROR
FINAL
FINAL ERROR MESSAGE
END OF JOB
TAPE ERROR?
12
NO
FILLT
I=I-1
YES
IS I GREATER THAN 0?
NO
SAVE NO. OF ACTIVE TAPES
IS IT ZERO?
YES
10A
NO
8

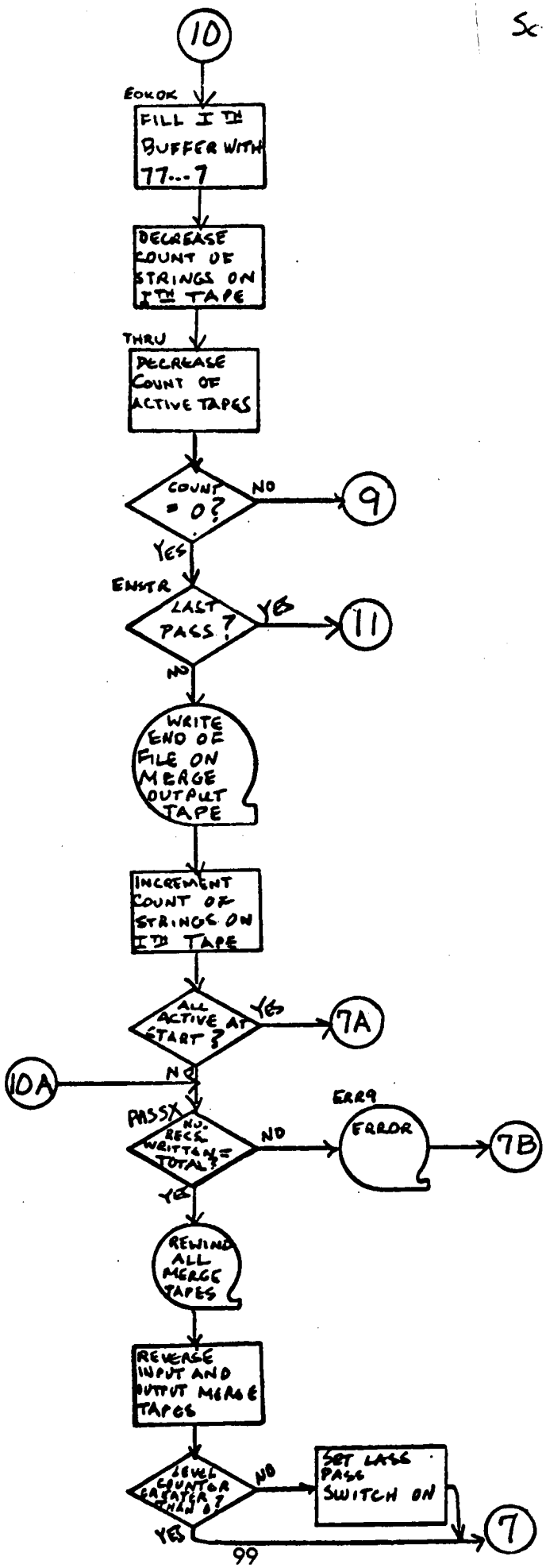
CR-62021


8
AREA
ZERO OUT PREVIOUS RECORD
LAST PASS?
YES
NO
TAPEK
IS THERE A NEXT MERGE TAPE?
YES
NO
INCREMENT MINIMUM NUMBER OF STRINGS ON A MERGE TAPE (LEVEL)
GET FIRST MERGE TAPE
SAVEK
STORE IN MERGEØ
9

CR-62021

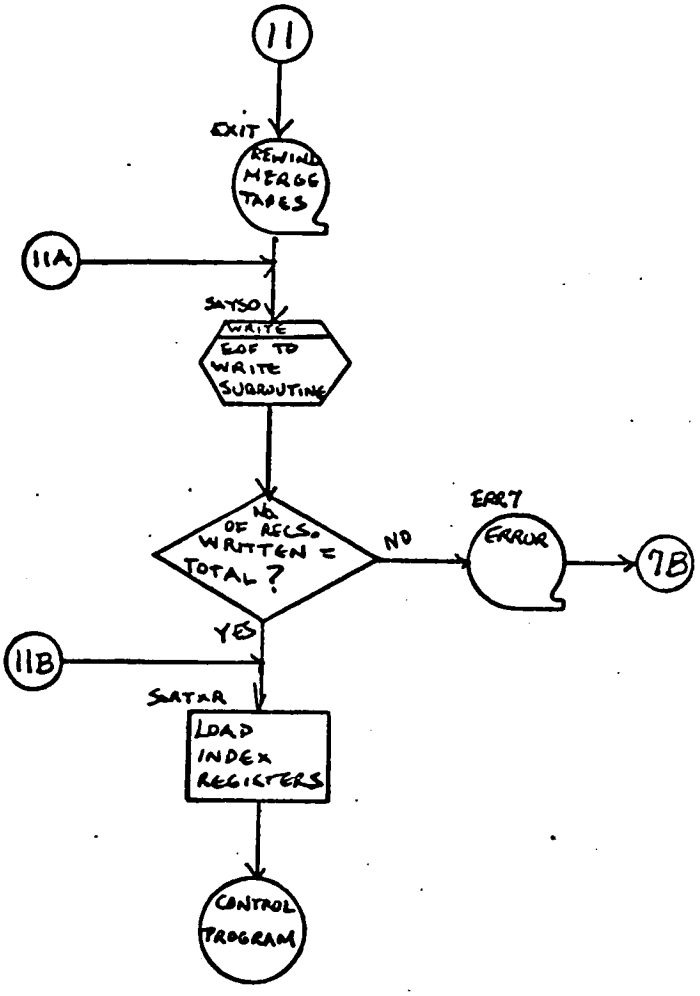

9
FIND
M=MERGE ORDER AND X=M-1
GET2
N=X
COMPR
COMP
IS M TH REC. GREATER THAN N TH?
YES
NEW1
X=X-1
NO
TEMP
X=X-1
X GREATER THAN 0
YES
M=X
NO
I=N
X GREATER THAN 0?
YES
NO
I=M
FOUND
COMP
IS I TH RECORD SMALLER THAN PREVIOUSLY WRITTEN RECORD
YES
ERR8
ERROR
7B
NO
NEWPR
MOVE I TH RECORD TO PREVIOUS STORAGE
LAST PASS?
YES
POST
WRITE OUTPUT I TH RECORD
READ
INCREMENT COUNT OF RECORDS WRITTEN
READ RECORD FROM I TH TAPE INTO BUFF(I)
NO
WRITE I TH REC. ON MERGE OUTPUT TAPE
12
YES
TAPE ERROR
NO
YES
END OF FILE
NO
10

CR-62021

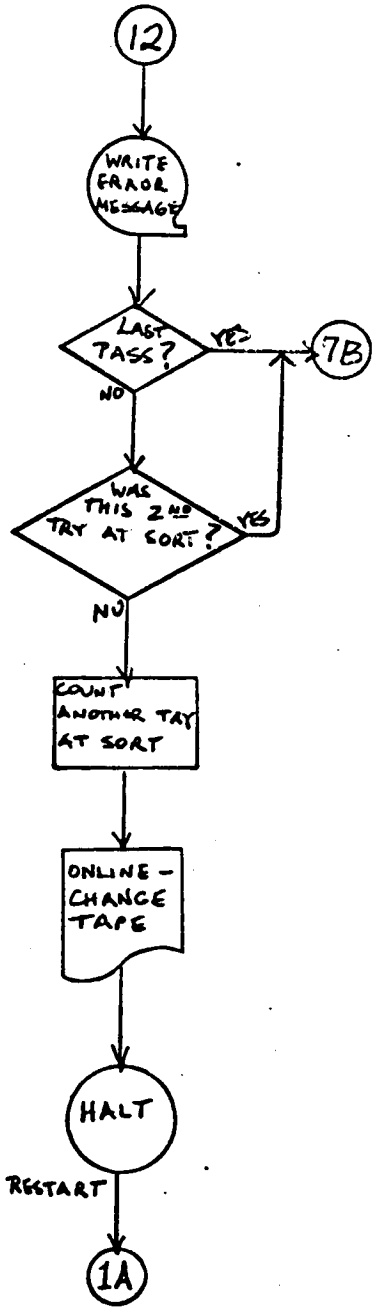


CR-6202

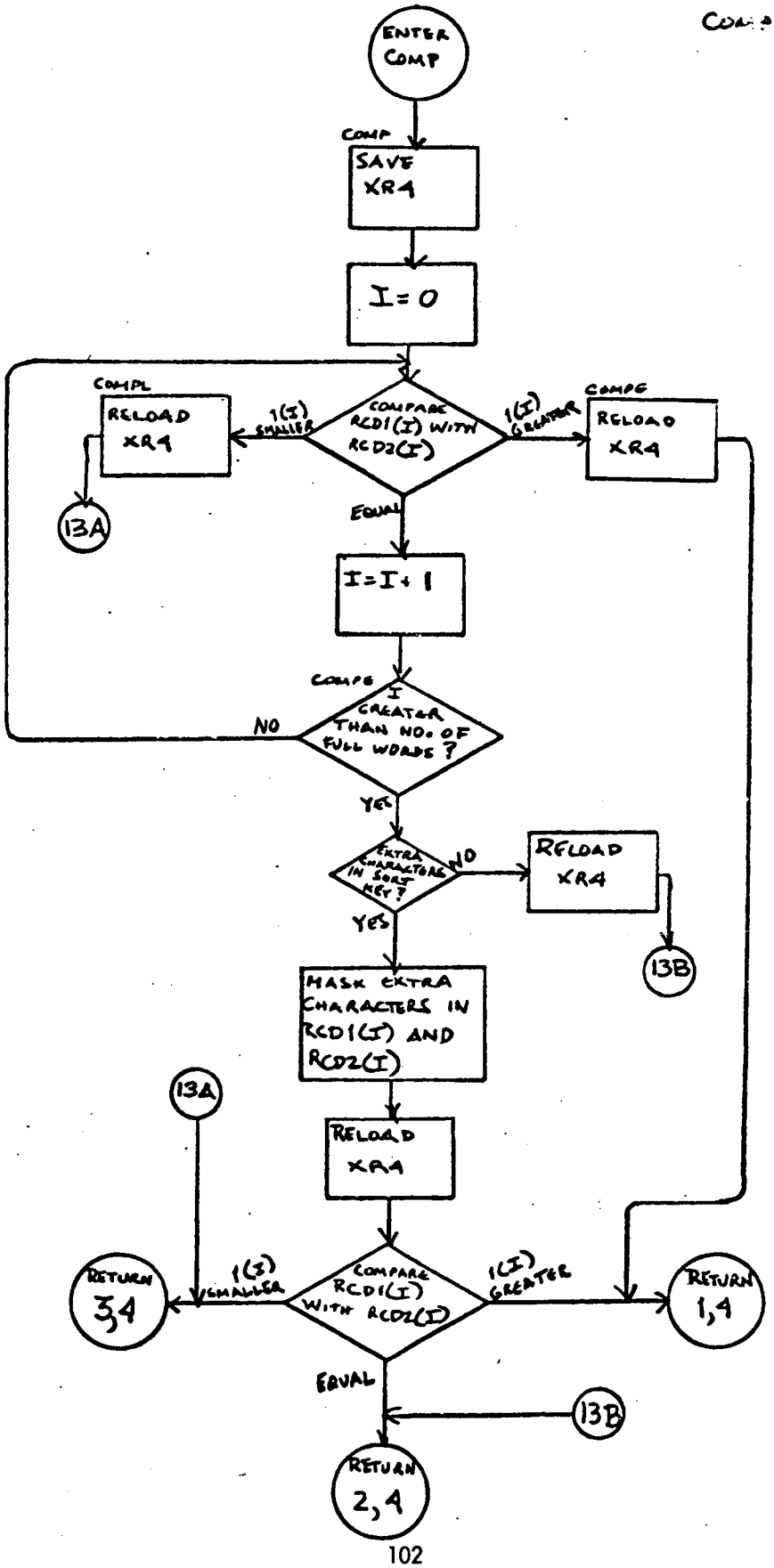
1 to Short


11
EXIT
REWIND MERGE TAPES
11A
SAYSO
WRITE EOF TO WRITE SUBROUTINE
NO. OF RECS. WRITTEN = TOTAL ?
NO
ERR7
ERROR
7B
YES
11B
SATAR
LOAD INDEX REGISTERS
CONTROL PROGRAM

CR-62021

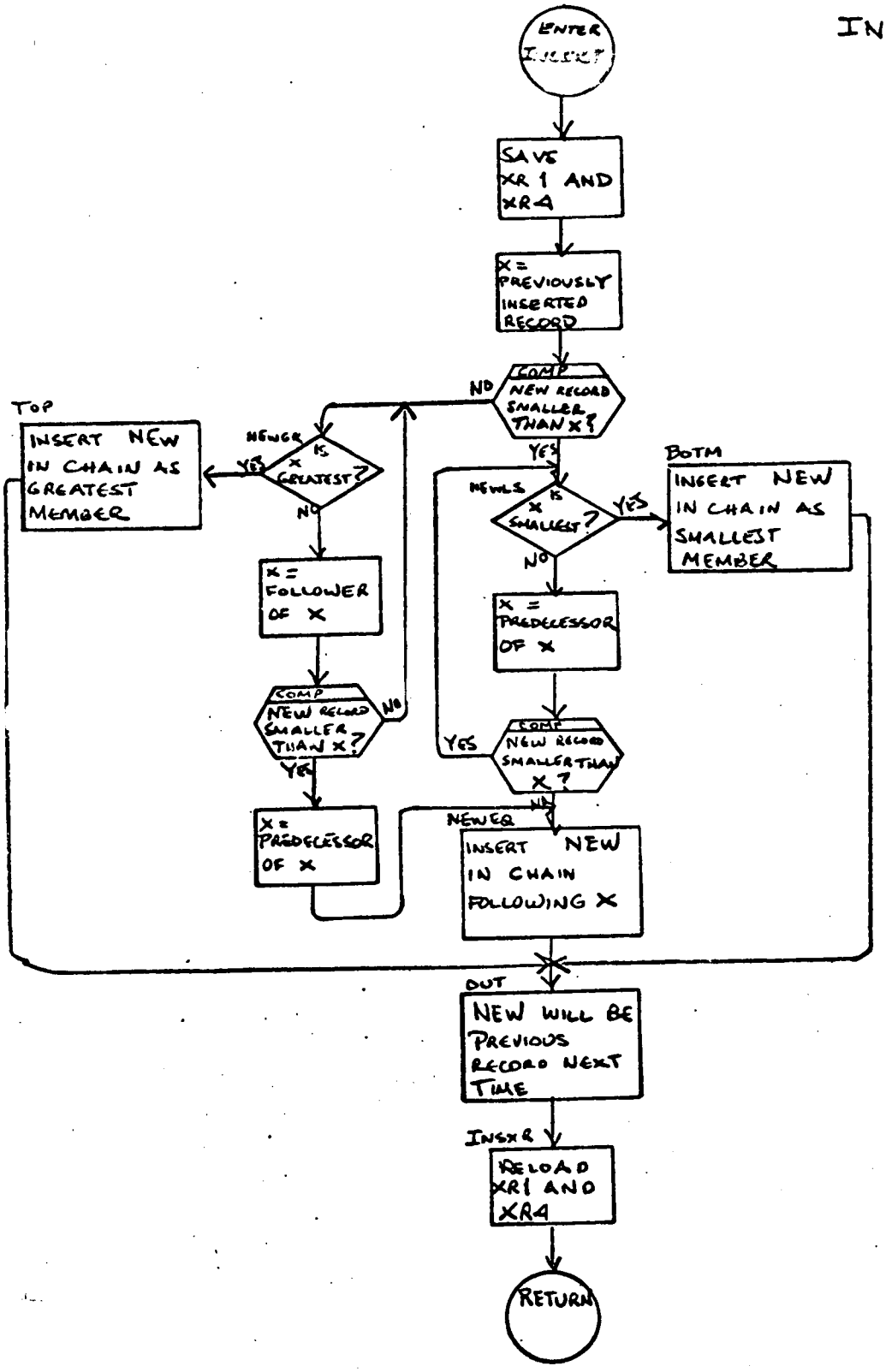

12
WRITE ERROR MESSAGE
LAST PASS?
YES
7B
NO
WAS THIS 2ND TRY AT SORT?
YES
NO
COUNT ANOTHER TRY AT SORT
ONLINE - CHANGE TAPE
HALT
RESTART
1A

CR-62021


ENTER COMP
COMP
SAVE XR4
I = 0
COMPARE RCD1(I) WITH RCD2(I)
I(I) SMALLER
COMPL
RELOAD XR4
13A
I(I) GREATER
COMPG
RELOAD XR4
EQUAL
I = I + 1
COMPG
I GREATER THAN NO. OF FULL WORDS ?
NO
YES
EXTRA CHARACTERS IN SORT KEY ?
NO
RELOAD XR4
13B
YES
MASK EXTRA CHARACTERS IN RCD1(I) AND RCD2(I)
13A
RELOAD XR4
COMPARE RCD1(I) WITH RCD2(I)
I(I) SMALLER
RETURN 3,4
I(I) GREATER
RETURN 1,4
EQUAL
13B
RETURN 2,4

CR 62021

INSERT SUBROUTINE



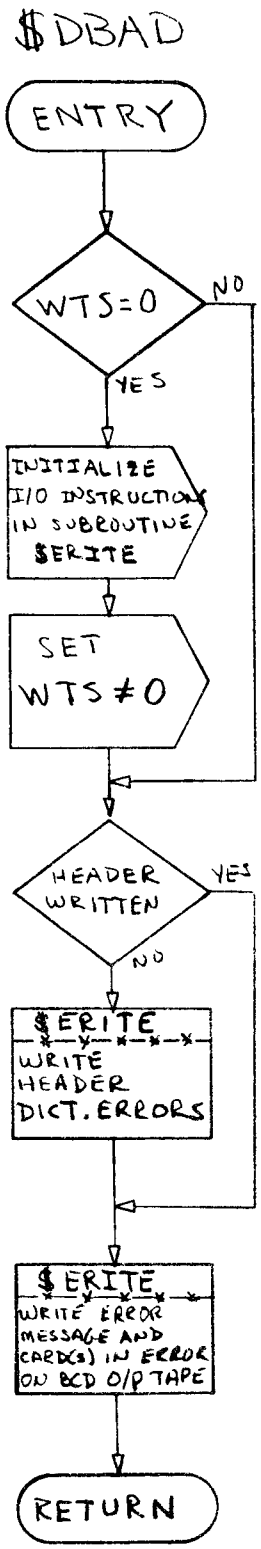
CR-62021

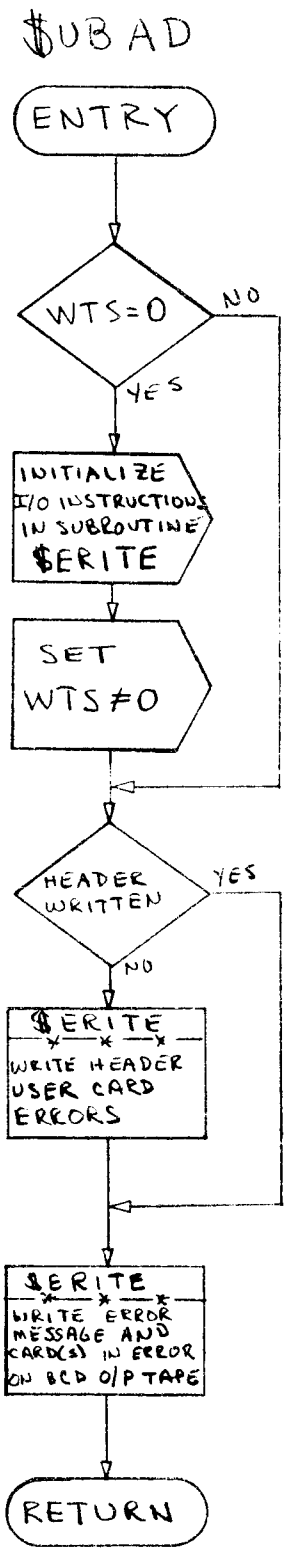
Subroutines UBAD and DBAD

UBAD is entered when an illegal user card is found; DBAD is entered when an illegal vocabulary card is found. Whichever subroutine of these two is entered first will initialize all output instructions in $ERITE.

Every TSX to UBAD or DBAD contains a number in its decrement that determines the error message to be written. This number will be placed in index register one. The RCH instruction effective address of the RCH will point to the I/O command that will write the error message desired.

CR-62021


$DBAD
ENTRY
WTS=0
NO
YES
INITIALIZE I/O INSTRUCTIONS IN SUBROUTINE $ERITE
SET WTS≠0
HEADER WRITTEN
YES
NO
$ERITE
WRITE HEADER DICT. ERRORS
$ERITE
WRITE ERROR MESSAGE AND CARD(S) IN ERROR ON BCD O/P TAPE
RETURN


$UBAD
ENTRY
WTS=0
NO
YES
INITIALIZE I/O INSTRUCTIONS IN SUBROUTINE $ERITE
SET WTS≠0
HEADER WRITTEN
YES
NO
$ERITE
WRITE HEADER USER CARD ERRORS
$ERITE
WRITE ERROR MESSAGE AND CARD(S) IN ERROR ON BCD O/P TAPE
RETURN

CR-62021

Subroutine UPDATE

Subroutine UPDATE is entered in phase 4. The purpose of UPDATE is to create or update the vocabulary tape while at the same time forming document and/or user profile tapes. This is accomplished by processing information from tape A (input sorted according to coded values).

The operation of the subroutine UPDATE can be discussed for two types of cases: (1) there is no old vocabulary so a new vocabulary is being created from document profile descriptors three words or less in length, or (2) there is an old vocabulary which has to be updated.

Case 1

A calculation of the number of logical records per file is made and blank space is provided for the first file of the new dictionary. Then a record is read from tape A using the subroutine READA.

If the descriptor is illegal it is rejected with an appropriate message on the BCD output tape. All messages on the BCD tape are handled by the subroutine WROUT. For valid descriptors, the type of descriptor is determined and the descriptor is processed accordingly. The possible types of descriptors are as follows:

1. Headers for either documents or users
2. Dictionary changes
    a. Add
    b. Delete
    c. Equate
        i. Primary (T in column 71 of input card)
        ii. Secondary (E in column 71 of input card)
    d. Trouble term
3. Document add descriptors
4. User descriptors
    a. Add
    b. Delete

For either document or user headers the record is sent to the routine MVAD which arranges the record in the format necessary for tapes B and C. Then the record is written out on the appropriate tape (six logical records per block).

For a vocabulary add card, a check is made to see if the descriptor has been added by a previous card. If not, the count of dictionary adds is increased by one, the descriptor is set up in the new dictionary by the subroutine FORM and a BCD message indicating the addition is made on the output tape. The vocabulary tape has a blocking factor of 15, and when the output buffer is full the tape is written by the subroutine WDICT.

If the descriptor is already on the vocabulary, a check is made on the first word of the 20-word alphabetic to see if it is blank. If blank, the alphabetic from the present descriptor is moved into the vocabulary record and a message is printed indicating this; if the alphabetic is not blank, a check is made to determine that the first words of the two alphabetic descriptors match. If they match, a message is printed saying that the descriptor is already on the vocabulary; otherwise a message is printed saying that the coded values are equal but the alphabetics differ. In both cases the descriptor is not added to the vocabulary.

Vocabulary delete records, while valid in an update run, are not allowed in a run to create the vocabulary. A message to this effect is put out on the BCD tape.

For vocabulary equate cards, processing varies depending on whether the descriptor is a primary or a secondary descriptor. If a primary descriptor does not match a descriptor previously added to the vocabulary, the descriptor is set up on the new vocabulary and the count of vocabulary adds is increased by one. A message indicating this action is printed. If there is a match, the descriptor already on the vocabulary is checked to see if it is a base descriptor, and if it is not, a message indicating this is printed. For a secondary descriptor, a check is also made to see if it is already on the vocabulary. If not, the count of vocabulary is increased, a message indicating an addition to the vocabulary is printed and the record is set up on the new vocabulary tape. The second word of the vocabulary record in this case will contain the primary coded value associated with this descriptor and the count of equate cards will be increased by one. If there is a match on the vocabulary, the count of equates will be increased and the primary coded value associated with the descriptor will be moved into the second word of the vocabulary record. A message will be printed indicating that this word has been changed.

User profiles must be corrected to reflect the synonym relationship established by vocabulary equate changes, hence each such change creates a dummy update for user profiles. Thus user profile tapes must be mounted whenever vocabulary equate changes are to be processed. No more than N/2 equate changes should be processed during a single computer run, where N = core storage required by subroutine UPDATE, COMMON storage excluded.

For a trouble term, a check is made to see if the descriptor has already been put on the vocabulary. If not, the count of vocabulary adds is increased, the record is set up on the vocabulary with the second word of the record set equal to zero, and a message indicating the addition of a trouble term is printed. If there is a match, the second word of the vocabulary record is set equal to zero and a message indicating that the descriptor was changed to a trouble term is printed.

For document cards, a check is made to see if the descriptor is on the vocabulary. If not, the descriptor is checked to see if it should be added. If it is not to be added, the descriptor, after being rearranged, is written out on the document tape by the subroutine WDOCT, which uses a blocking factor of six. If the descriptor is to be added to the vocabulary, the count of document adds is increased, a message indicating an addition was made to the vocabulary is printed and the record is set up in the vocabulary. A bit is set in the document record indicating that the descriptor is on the vocabulary, the record is rearranged and is written on the document tape. If the descriptor is already on the vocabulary, the dictionary record is updated and the bit is set in the document record indicating it is on the vocabulary. Then, as in the previous instance, the document record is rearranged and written out onto the document tape.

There are two kinds of user descriptors - user adds and user deletes. A user add is handled

in the same way as a document descriptor except the record is put out onto the user tape by the subroutine WUSE and there is a count for new user additions and a count for additions to old user profiles. User deletes are checked to see if there is a match on the vocabulary. If not, the record is rearranged and written out on the USER tape. If there is a match, the number of users having this descriptor is decreased on the vocabulary record and the record then is handled the same as is a document descriptor having a match on the vocabulary.

Whenever a coded descriptor is to be added to a profile (document or user), the original code is replaced by word 2 of the corresponding vocabulary record. Thus terms previously designated as synonyms or trouble terms are corrected for profile entry (or rejection).

While the descriptor from tape A is processed, an account is kept of the number of descriptors put on the vocabulary so that each file will have approximately 1/100th of the total number of descriptors. Also, a catalog is made of the minimum and maximum coded values for each file. In the situation where 100 files have not been written and tape A is finished, the remaining files will each contain 450 words of zeros and have maximum and minimum values consisting of computer words of octal sevens. When all descriptors have been processed the vocabulary tape is rewound and the first file record contains the tape label. The second record contains the catalog of minimum and maximum entries and the third record has a table of entries containing the number of logical records per file. This file is written by the subroutine WDLBL. Also, if there is anything in the buffers for the document and user tapes, these tapes are written. Finally, a summary of all the changes made to the vocabulary is written on the system output tape.

Case 2

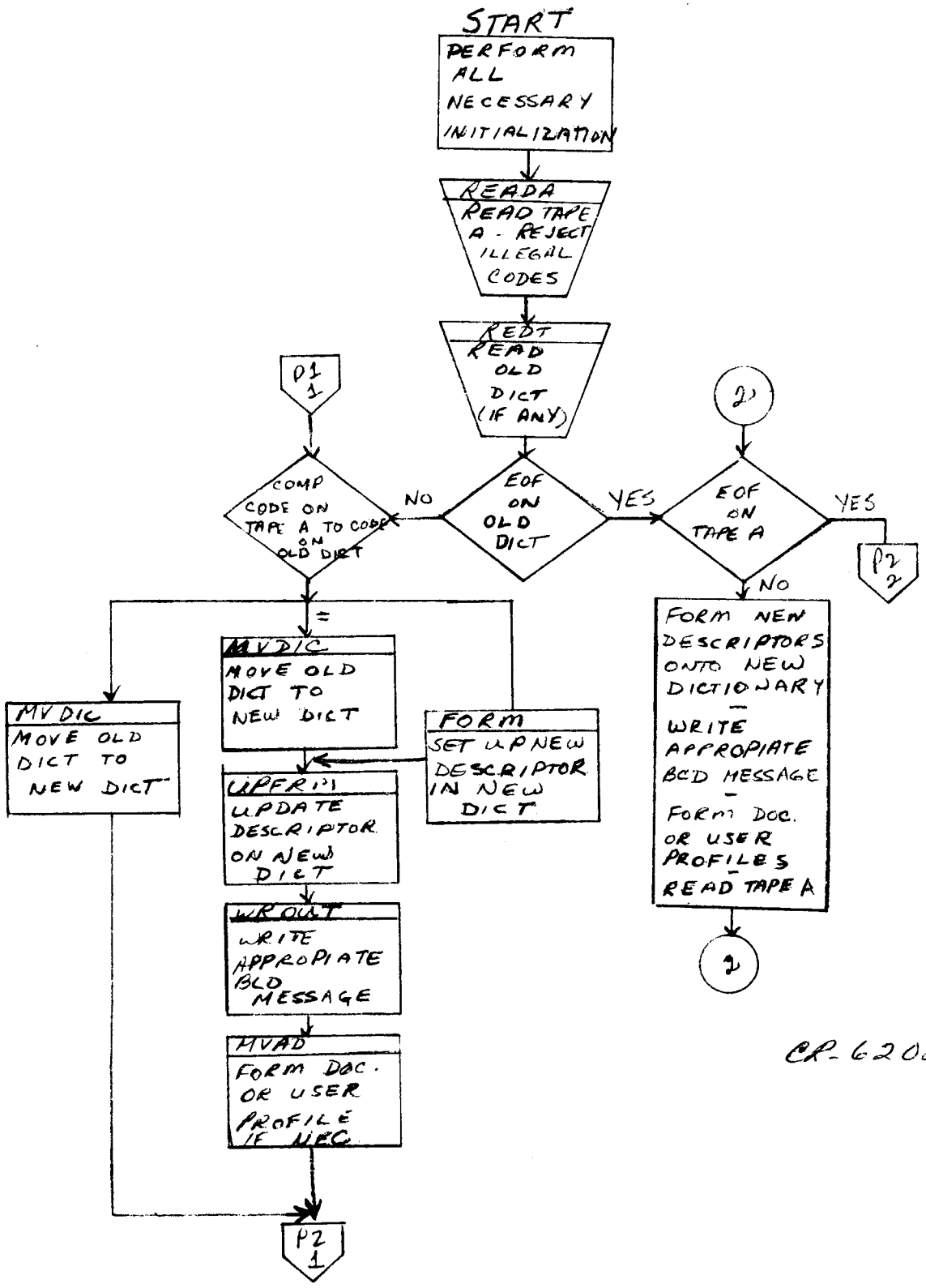
In a run to update the vocabulary, all descriptor records have to be matched against the old vocabulary. Until a match is found, the records read from the old vocabulary by the subroutine REDT are written out onto the new vocabulary. The size of each file in the new vocabulary will depend, of course, on both the number of descriptors on tape A and the number of descriptors on the old vocabulary. Also, the maximum and minimum coded values are again provided for each file on the new vocabulary.

In general, for an updating run, the types of descriptors are processed in the same way as in a run to create the vocabulary, except that dictionary delete descriptors are allowed. The descriptor is checked for a match on the old vocabulary or for a match with a descriptor just added to the vocabulary. If there is no match, a message indicating that the descriptor is not on the vocabulary is printed. If there is a match, the count of vocabulary deletes is increased, the record is deleted from the vocabulary and a message is printed saying the descriptor has been deleted.

If all processing of tape A is done but there still remain records on the old vocabulary, these records will be written from the old vocabulary onto the new vocabulary. On the other hand, if the old vocabulary is finished before tape A, the remaining descriptors will be processed from tape A as described in Case 1. In either situation, final processing of the new vocabulary tape and the document and user tapes is accomplished in the same manner as in Case 1.

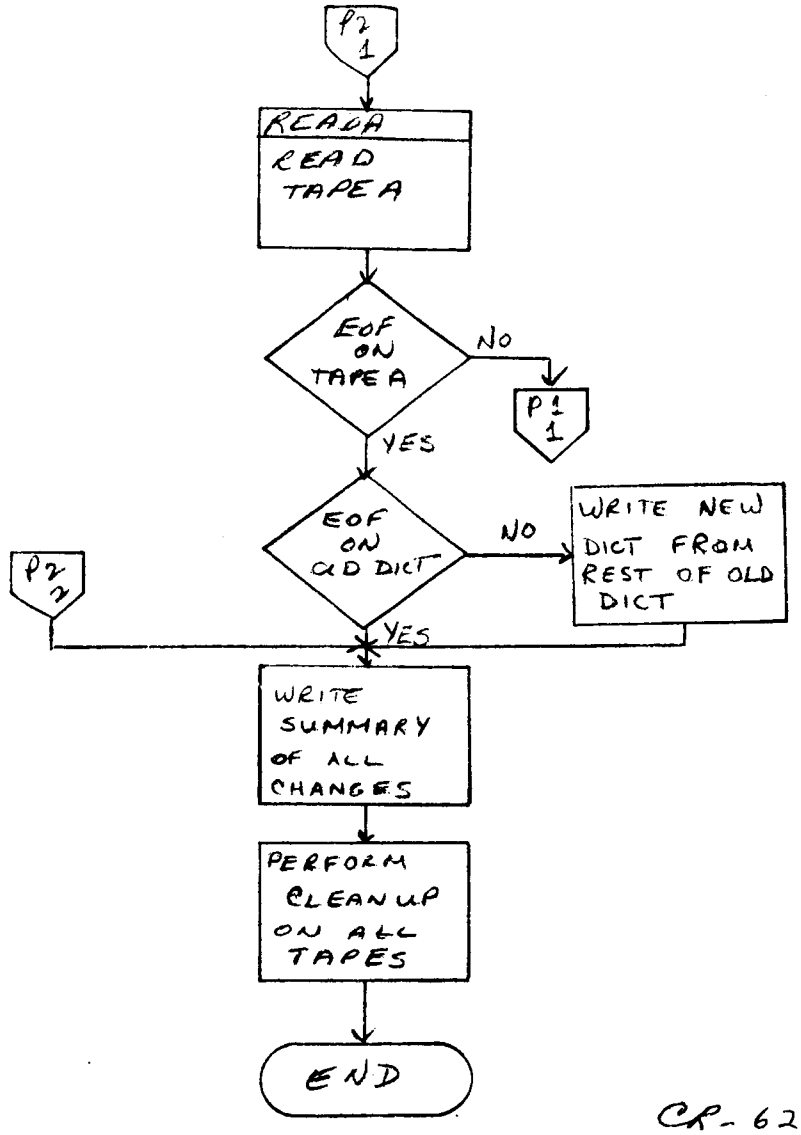
CR-62021

Block Diagram

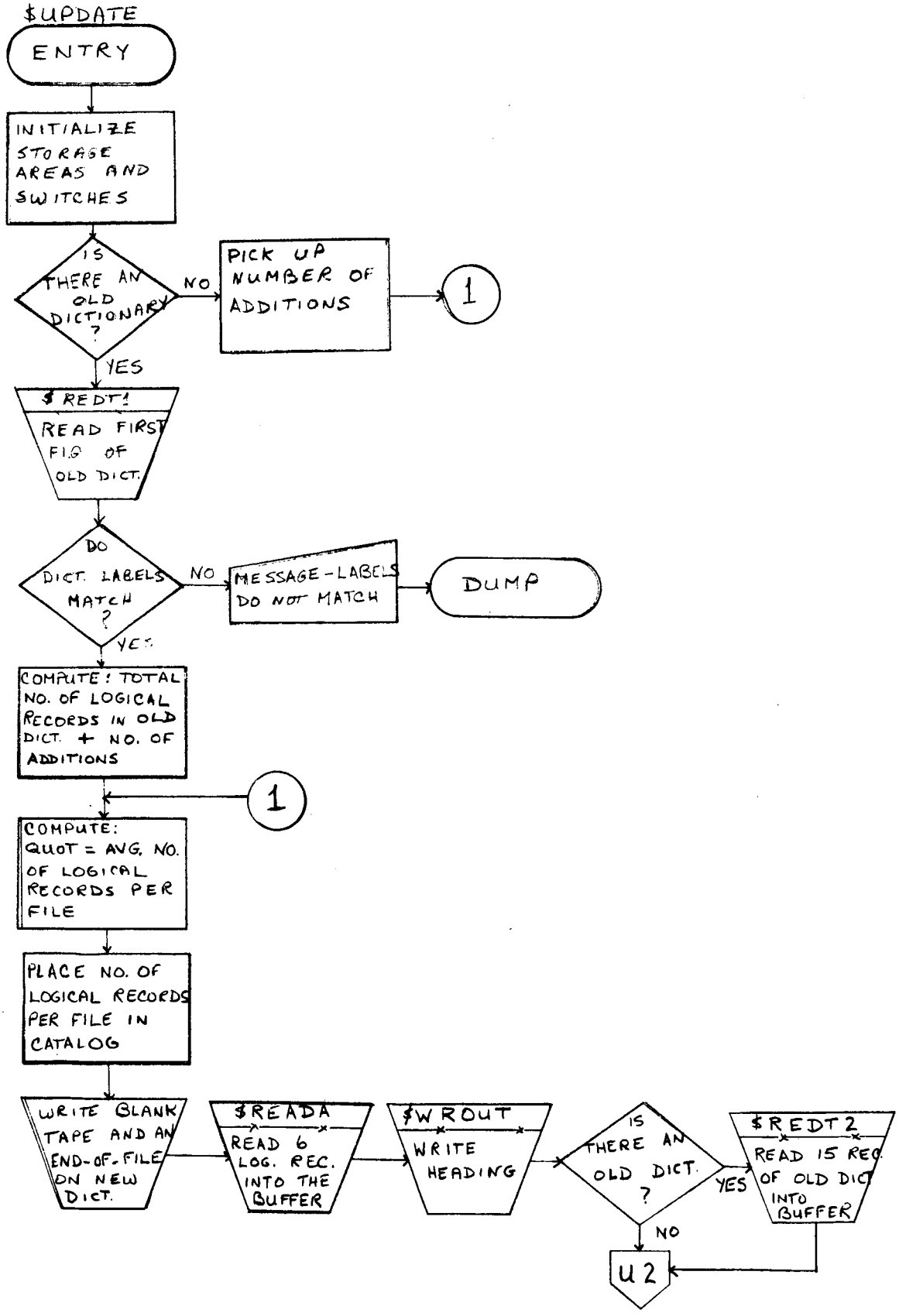


CR-6202/

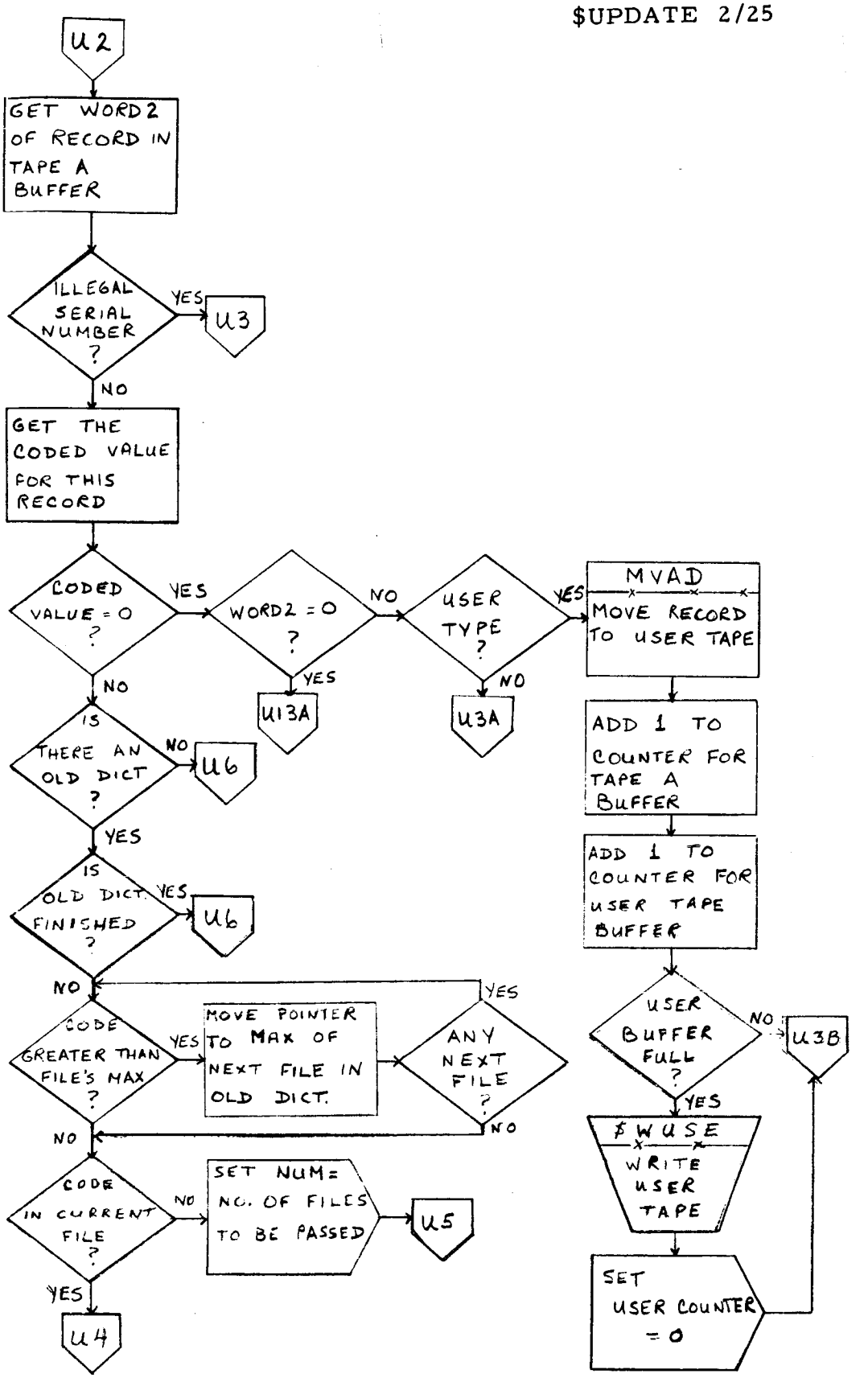
Block Diagram



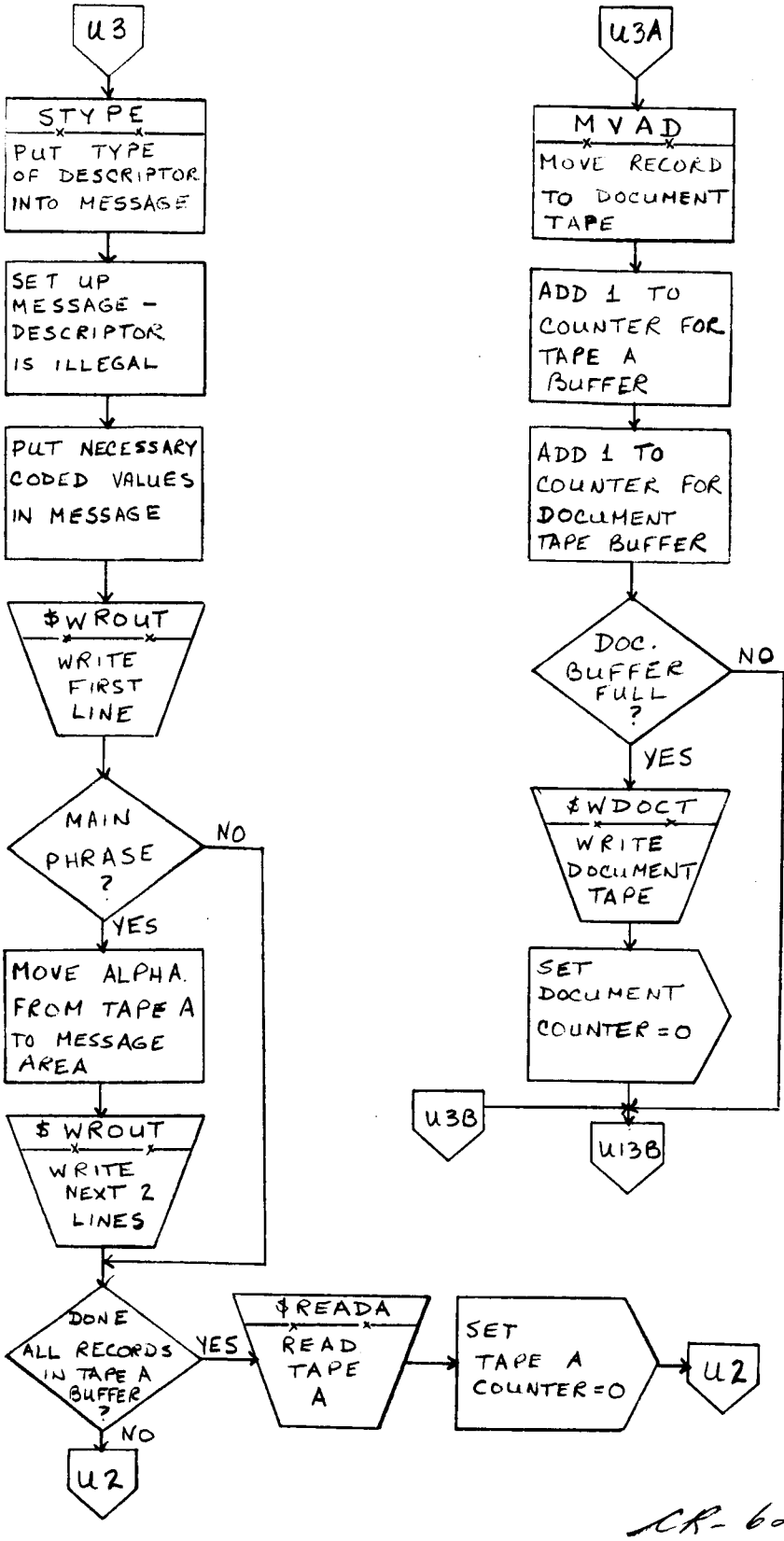
CR-62021


$UPDATE
ENTRY
INITIALIZE STORAGE AREAS AND SWITCHES
IS THERE AN OLD DICTIONARY ?
NO
PICK UP NUMBER OF ADDITIONS
1
YES
$REDT1
READ FIRST F.G. OF OLD DICT.
DO DICT. LABELS MATCH ?
NO
MESSAGE - LABELS DO NOT MATCH
DUMP
YES
COMPUTE: TOTAL NO. OF LOGICAL RECORDS IN OLD DICT. + NO. OF ADDITIONS
1
COMPUTE: QUOT = AVG. NO. OF LOGICAL RECORDS PER FILE
PLACE NO. OF LOGICAL RECORDS PER FILE IN CATALOG
WRITE BLANK TAPE AND AN END-OF-FILE ON NEW DICT.
$READA
READ 6 LOG. REC. INTO THE BUFFER
$WROUT
WRITE HEADING
IS THERE AN OLD DICT. ?
YES
$REDT2
READ 15 REC. OF OLD DICT INTO BUFFER
NO
U2

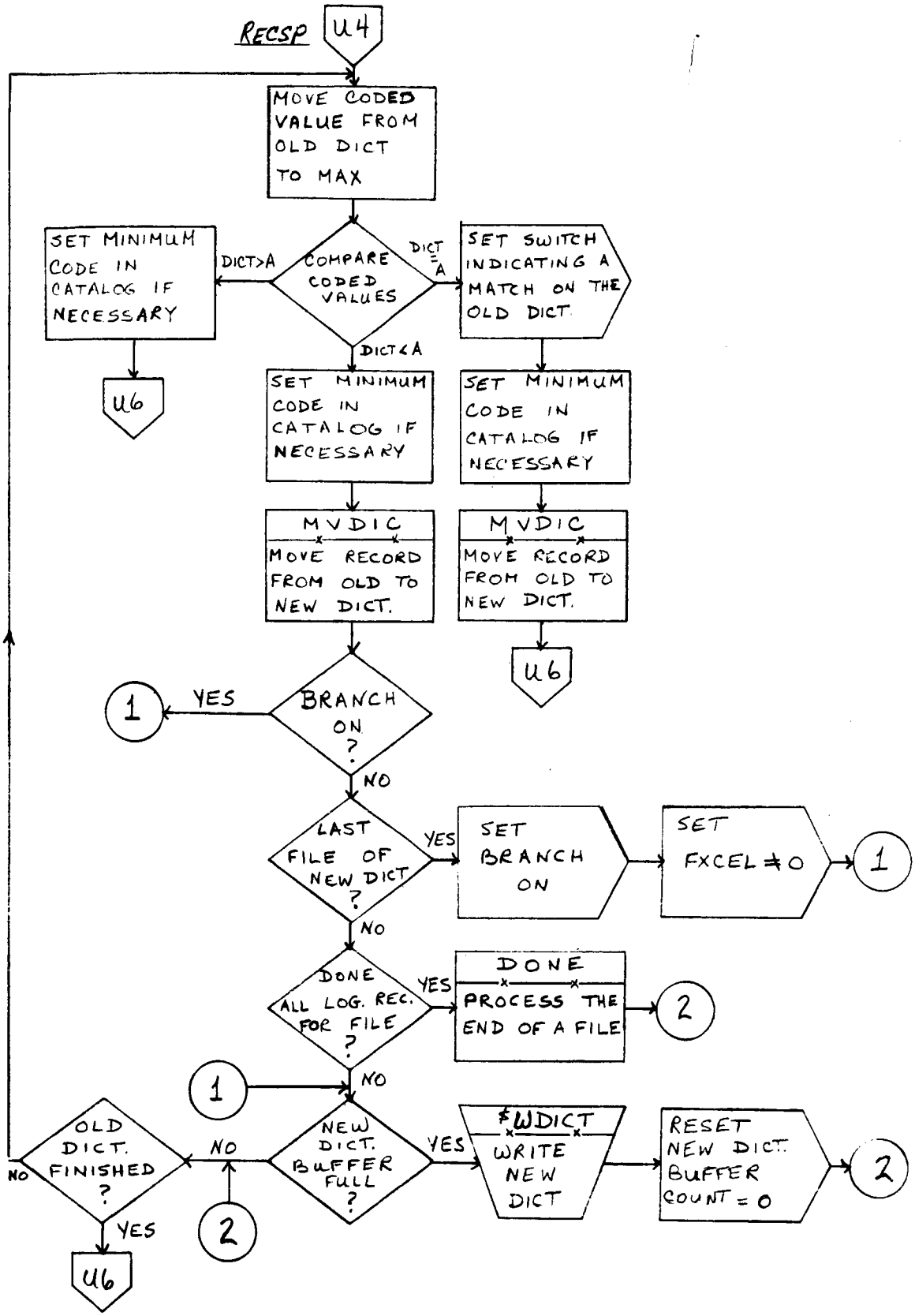
CR-62021


U2
GET WORD2 OF RECORD IN TAPE A BUFFER
ILLEGAL SERIAL NUMBER ?
YES
U3
NO
GET THE CODED VALUE FOR THIS RECORD
CODED VALUE = 0 ?
YES
WORD2 = 0 ?
NO
USER TYPE ?
YES
MVAD
MOVE RECORD TO USER TAPE
YES
U13A
NO
U3A
NO
IS THERE AN OLD DICT ?
NO
U6
YES
IS OLD DICT FINISHED ?
YES
U6
NO
CODE GREATER THAN FILE'S MAX ?
YES
MOVE POINTER TO MAX OF NEXT FILE IN OLD DICT.
ANY NEXT FILE ?
YES
NO
NO
CODE IN CURRENT FILE ?
NO
SET NUM= NO. OF FILES TO BE PASSED
U5
YES
U4
ADD 1 TO COUNTER FOR TAPE A BUFFER
ADD 1 TO COUNTER FOR USER TAPE BUFFER
USER BUFFER FULL ?
NO
U3B
YES
$WUSE
WRITE USER TAPE
SET USER COUNTER = 0

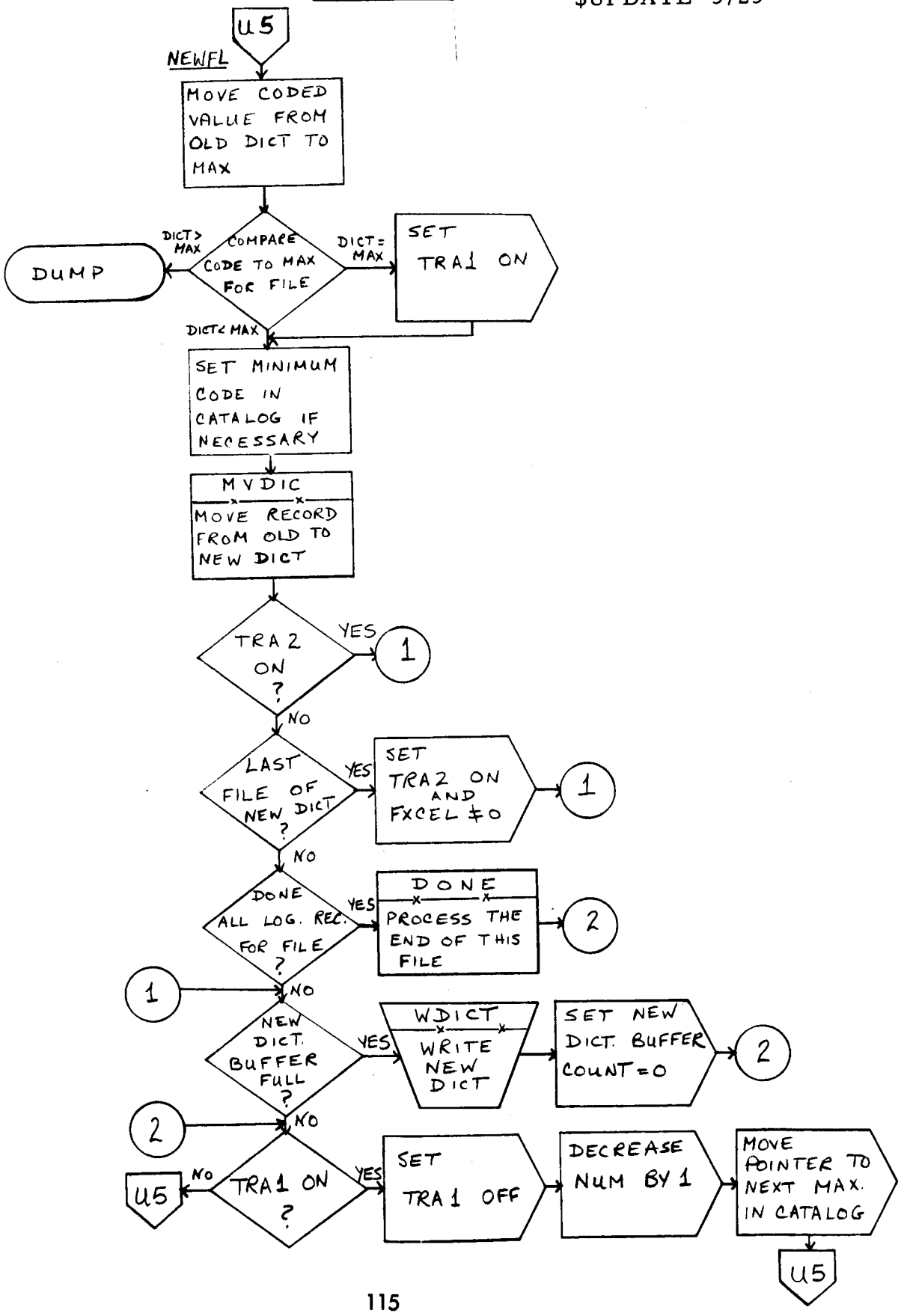
CR-62021


U3
STYPE
PUT TYPE OF DESCRIPTOR INTO MESSAGE
SET UP MESSAGE - DESCRIPTOR IS ILLEGAL
PUT NECESSARY CODED VALUES IN MESSAGE
$WROUT
WRITE FIRST LINE
MAIN PHRASE ?
NO
YES
MOVE ALPHA. FROM TAPE A TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
DONE ALL RECORDS IN TAPE A BUFFER ?
YES
NO
U2
$READA
READ TAPE A
SET TAPE A COUNTER=0
U2
U3A
MVAD
MOVE RECORD TO DOCUMENT TAPE
ADD 1 TO COUNTER FOR TAPE A BUFFER
ADD 1 TO COUNTER FOR DOCUMENT TAPE BUFFER
DOC. BUFFER FULL ?
NO
YES
$WDOCT
WRITE DOCUMENT TAPE
SET DOCUMENT COUNTER=0
U3B
U13B

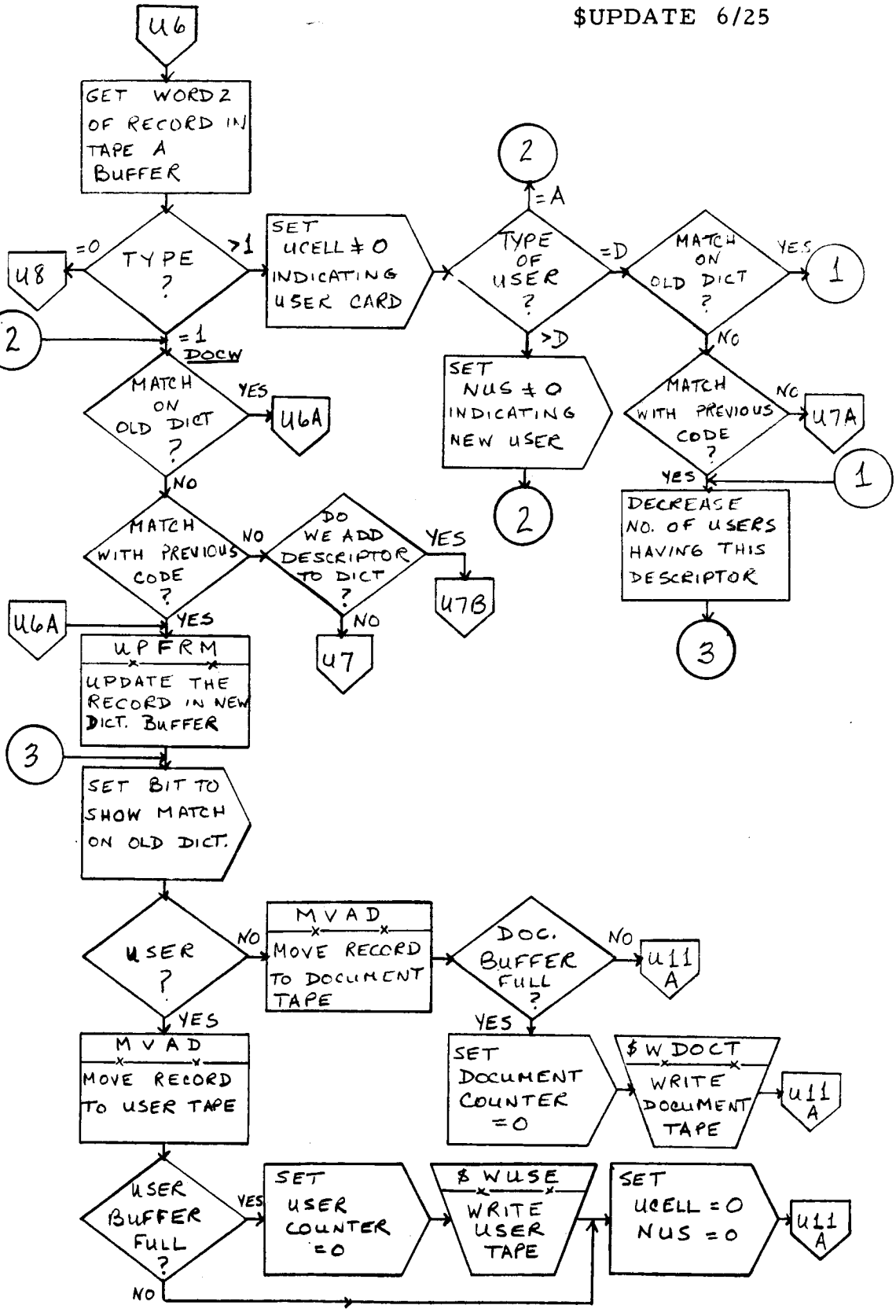
CR-62021


RECSP
U4
MOVE CODED VALUE FROM OLD DICT TO MAX
COMPARE CODED VALUES
DICT>A
DICT=A
DICT<A
SET MINIMUM CODE IN CATALOG IF NECESSARY
U6
SET SWITCH INDICATING A MATCH ON THE OLD DICT.
SET MINIMUM CODE IN CATALOG IF NECESSARY
MVDIC
MOVE RECORD FROM OLD TO NEW DICT.
U6
BRANCH ON ?
YES
1
NO
LAST FILE OF NEW DICT ?
YES
SET BRANCH ON
SET FXCEL≠0
1
NO
DONE ALL LOG. REC. FOR FILE ?
YES
DONE
PROCESS THE END OF A FILE
2
1
NO
NEW DICT. BUFFER FULL ?
YES
$WDICT
WRITE NEW DICT
RESET NEW DICT. BUFFER COUNT = 0
2
OLD DICT. FINISHED ?
NO
2
NO
YES
U6

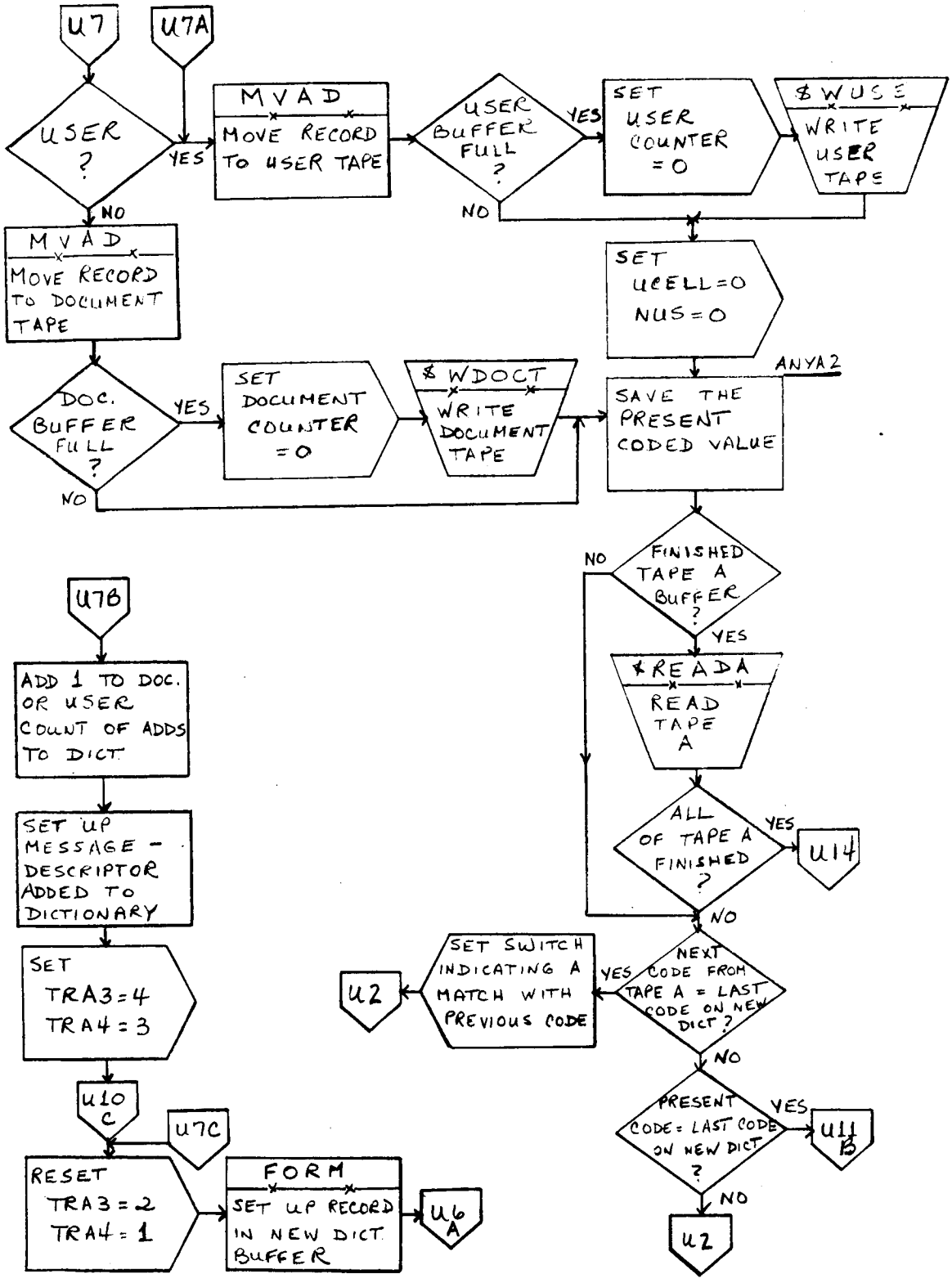
CR-62021


U5
NEWFL
MOVE CODED VALUE FROM OLD DICT TO MAX
COMPARE CODE TO MAX FOR FILE
DICT> MAX
DUMP
DICT= MAX
SET TRA1 ON
DICT< MAX
SET MINIMUM CODE IN CATALOG IF NECESSARY
MVDIC
MOVE RECORD FROM OLD TO NEW DICT
TRA 2 ON ?
YES
1
NO
LAST FILE OF NEW DICT ?
YES
SET TRA2 ON AND FXCEL ≠0
1
NO
DONE ALL LOG. REC. FOR FILE ?
YES
DONE
PROCESS THE END OF THIS FILE
2
1
NO
NEW DICT. BUFFER FULL ?
YES
WDICT
WRITE NEW DICT
SET NEW DICT. BUFFER COUNT=0
2
2
NO
U5
NO
TRA1 ON ?
YES
SET TRA1 OFF
DECREASE NUM BY 1
MOVE POINTER TO NEXT MAX. IN CATALOG
U5

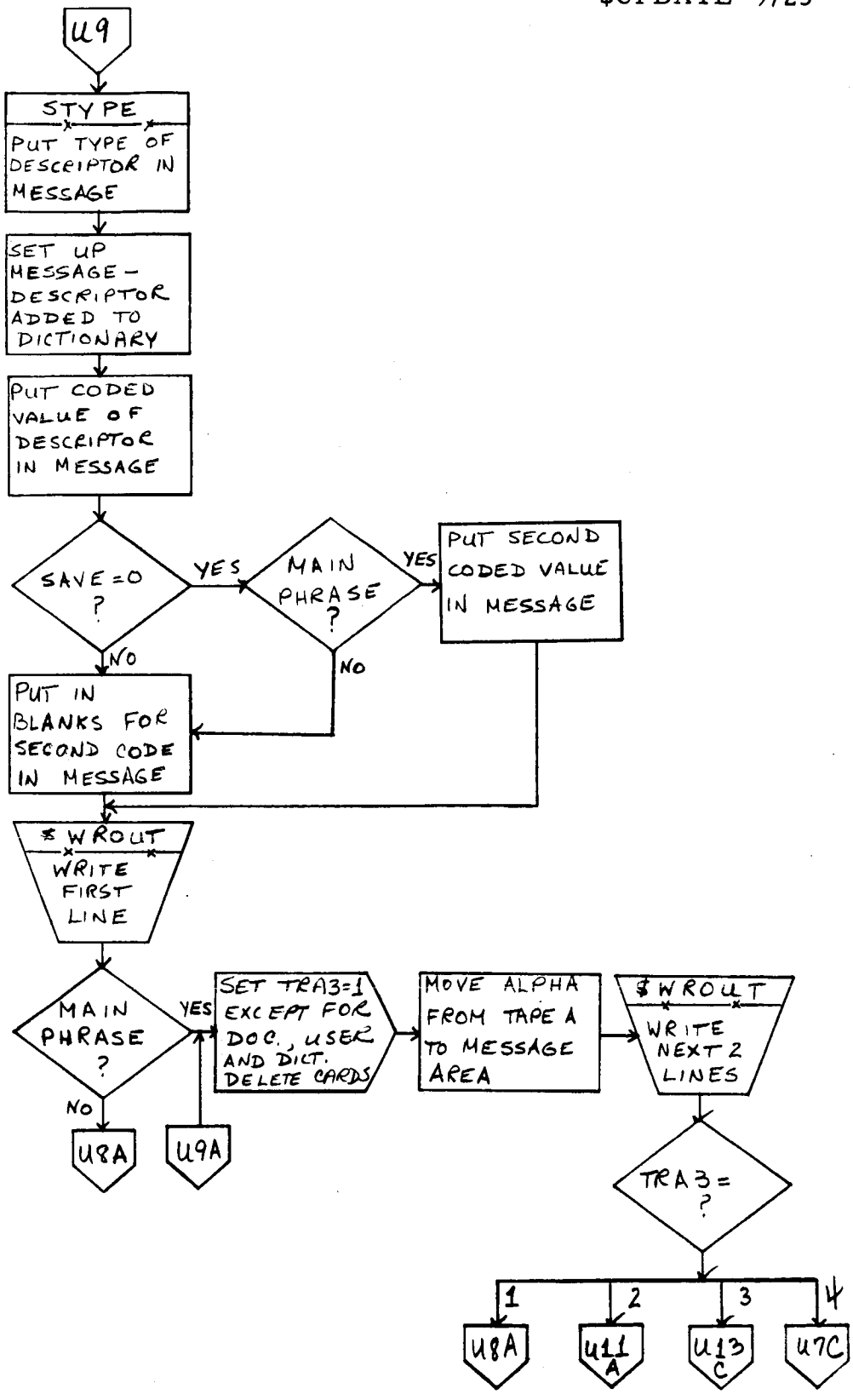
CR-62021


U6
GET WORD 2 OF RECORD IN TAPE A BUFFER
TYPE ?
=0
U8
>1
SET UCELL ≠ 0 INDICATING USER CARD
TYPE OF USER ?
=A
2
=D
MATCH ON OLD DICT ?
YES
1
>D
SET NUS ≠ 0 INDICATING NEW USER
NO
MATCH WITH PREVIOUS CODE ?
NO
U7A
YES
DECREASE NO. OF USERS HAVING THIS DESCRIPTOR
3
2
=1
DOCW
MATCH ON OLD DICT ?
YES
U6A
NO
MATCH WITH PREVIOUS CODE ?
NO
DO WE ADD DESCRIPTOR TO DICT ?
YES
U7B
NO
U7
U6A
YES
UPFRM
UPDATE THE RECORD IN NEW DICT. BUFFER
3
SET BIT TO SHOW MATCH ON OLD DICT.
USER ?
NO
MVAD
MOVE RECORD TO DOCUMENT TAPE
DOC. BUFFER FULL ?
NO
U11 A
YES
MVAD
MOVE RECORD TO USER TAPE
YES
SET DOCUMENT COUNTER =0
$WDOCT
WRITE DOCUMENT TAPE
U11 A
USER BUFFER FULL ?
YES
SET USER COUNTER =0
$WUSE
WRITE USER TAPE
SET UCELL = 0 NUS = 0
U11 A
NO

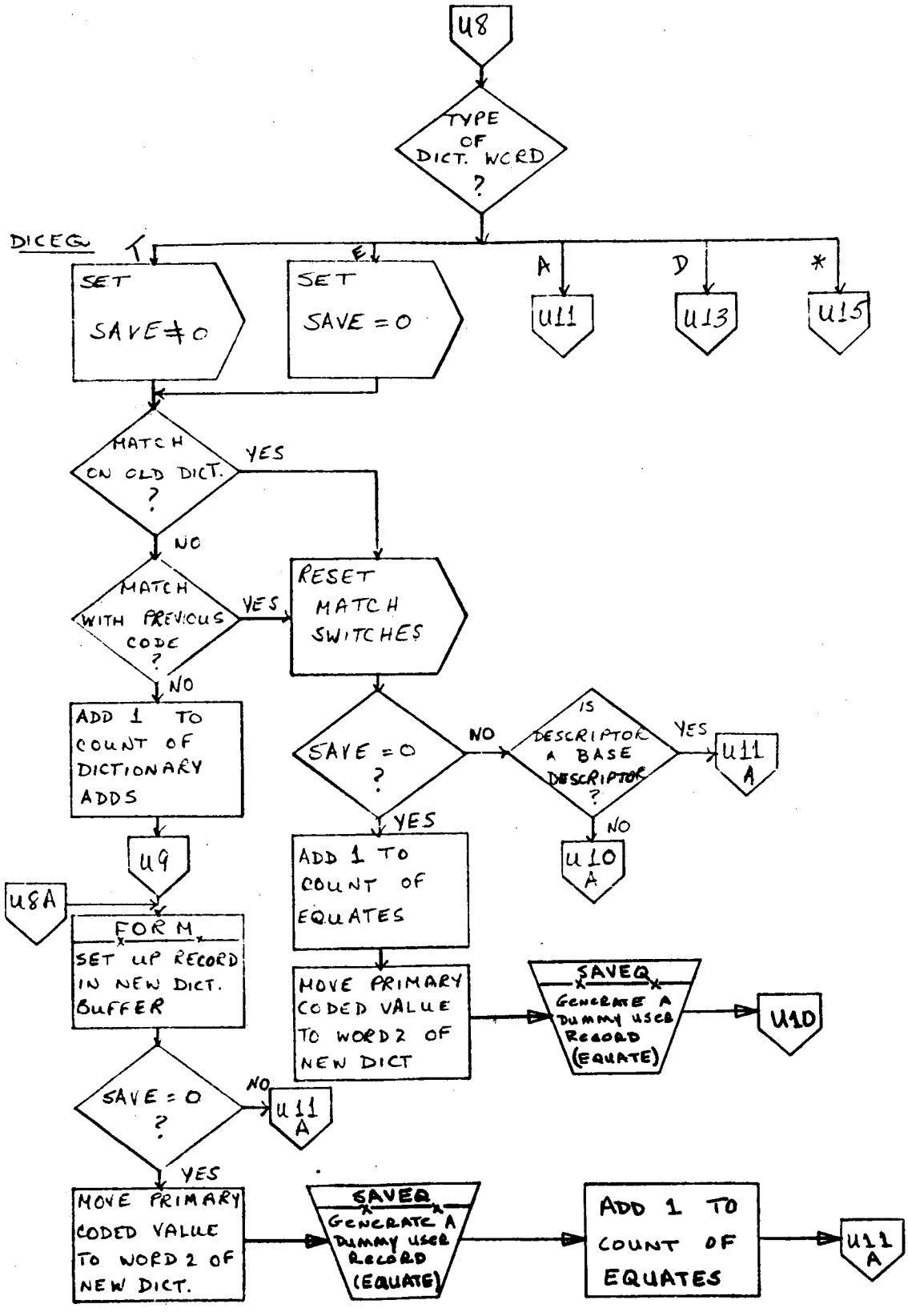
CR 62021


U7
U7A
USER ?
YES
NO
MVAD
MOVE RECORD TO USER TAPE
USER BUFFER FULL ?
YES
NO
SET USER COUNTER = 0
$WUSE
WRITE USER TAPE
MVAD
MOVE RECORD TO DOCUMENT TAPE
SET UCELL=0
NUS=0
DOC. BUFFER FULL ?
YES
NO
SET DOCUMENT COUNTER = 0
$WDOCT
WRITE DOCUMENT TAPE
ANYA2
SAVE THE PRESENT CODED VALUE
FINISHED TAPE A BUFFER ?
NO
YES
U7B
ADD 1 TO DOC. OR USER COUNT OF ADDS TO DICT
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
SET TRA3=4 TRA4=3
$READA
READ TAPE A
ALL OF TAPE A FINISHED ?
YES
U14
NO
NEXT CODE FROM TAPE A = LAST CODE ON NEW DICT ?
YES
SET SWITCH INDICATING A MATCH WITH PREVIOUS CODE
U2
NO
PRESENT CODE = LAST CODE ON NEW DICT ?
YES
U11 B
NO
U2
U10 C
U7C
RESET TRA3=2 TRA4=1
FORM
SET UP RECORD IN NEW DICT BUFFER
U6 A

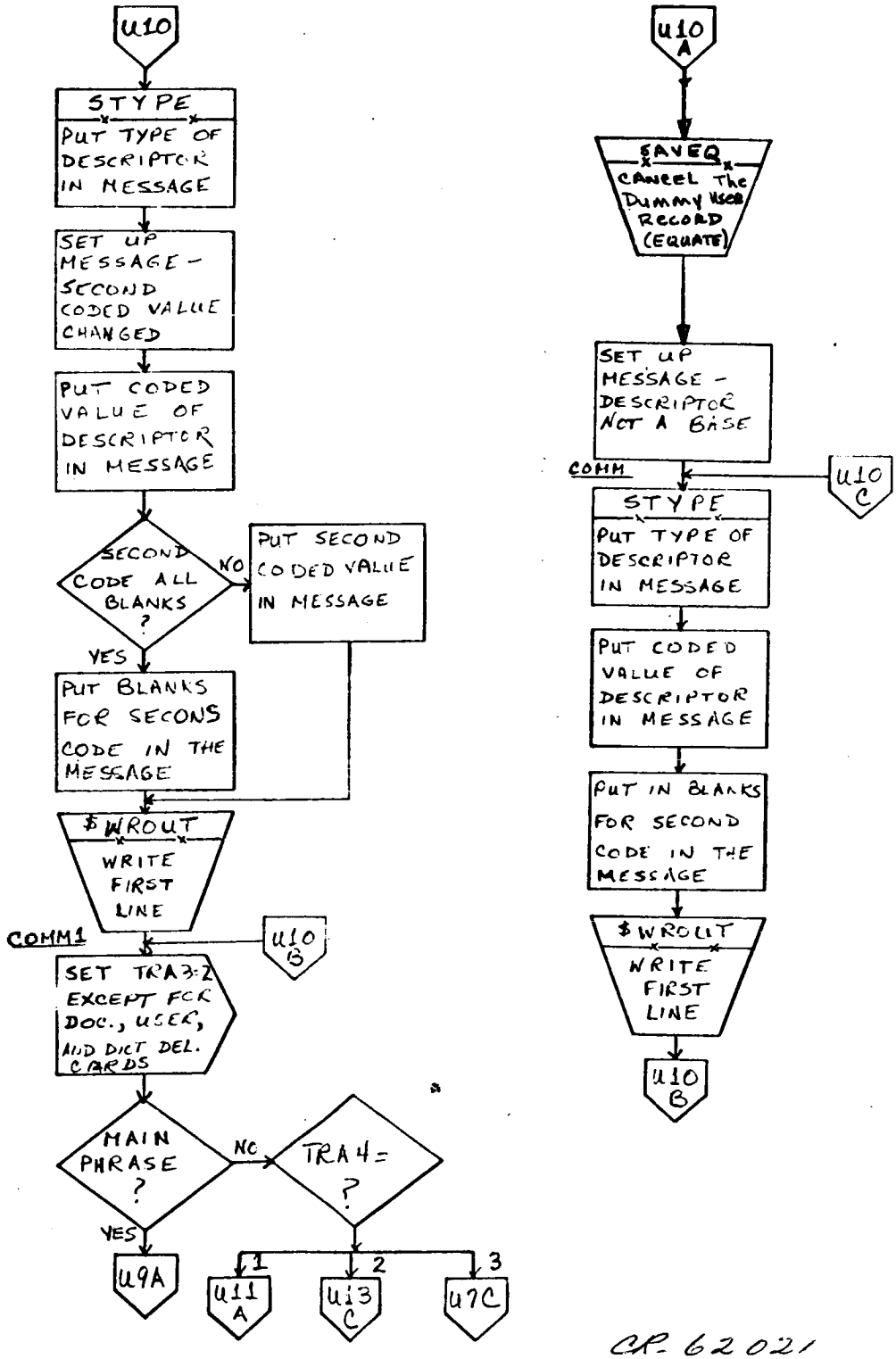
CR-62021


U9
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
SAVE=0 ?
YES
MAIN PHRASE ?
YES
PUT SECOND CODED VALUE IN MESSAGE
NO
NO
PUT IN BLANKS FOR SECOND CODE IN MESSAGE
$WROUT
WRITE FIRST LINE
MAIN PHRASE ?
YES
SET TRA3=1 EXCEPT FOR DOC., USER AND DICT. DELETE CARDS
MOVE ALPHA FROM TAPE A TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
NO
U8A
U9A
TRA3 = ?
1
2
3
4
U8A
U11 A
U13 C
U7C

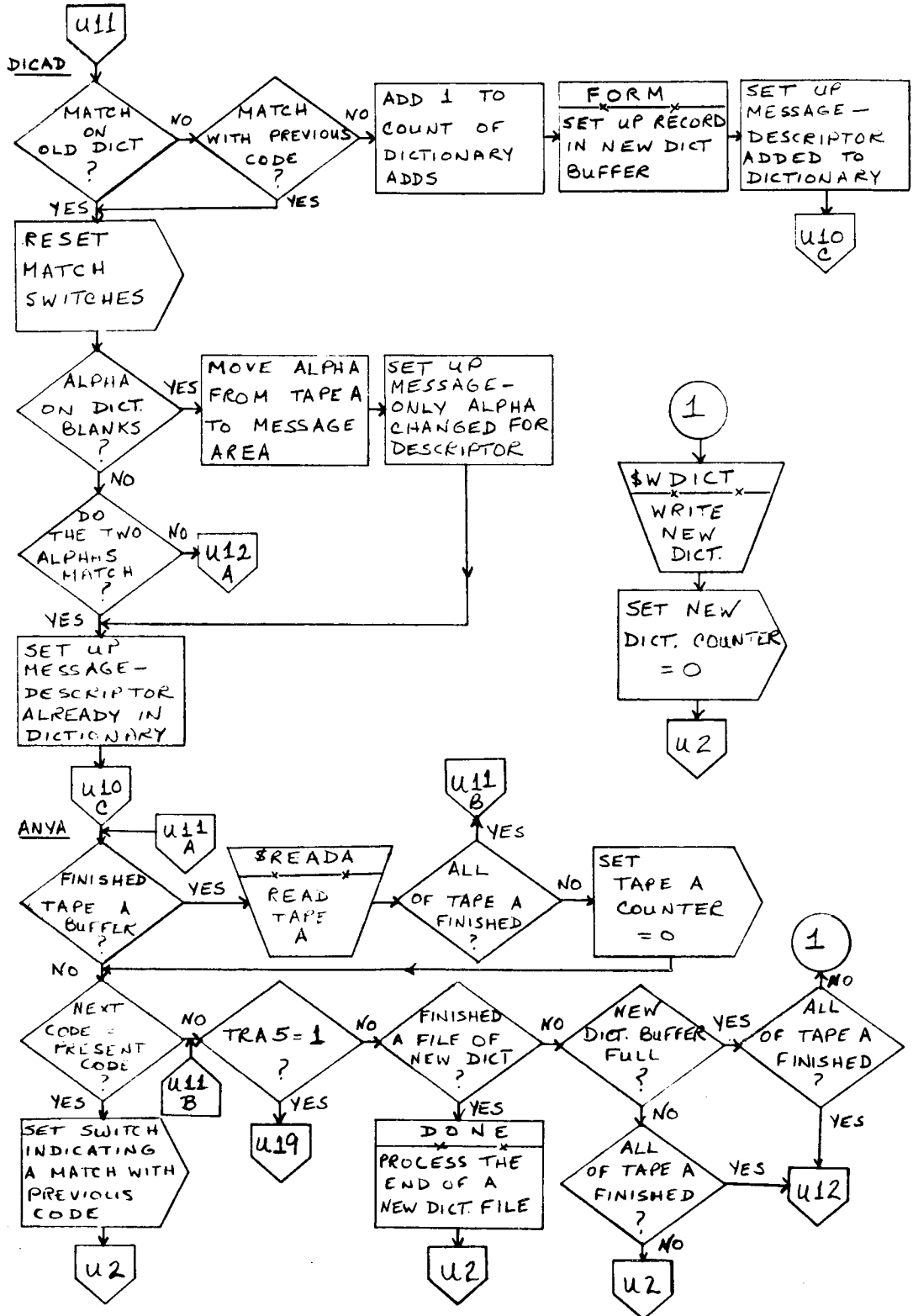
CR-6.2021


U8
TYPE OF DICT. WORD ?
DICEQU
T
E
A
D
*
SET SAVE≠0
SET SAVE=0
U11
U13
U15
MATCH ON OLD DICT. ?
YES
NO
MATCH WITH PREVIOUS CODE ?
YES
RESET MATCH SWITCHES
NO
ADD 1 TO COUNT OF DICTIONARY ADDS
SAVE=0 ?
NO
IS DESCRIPTOR A BASE DESCRIPTOR ?
YES
U11 A
NO
U10 A
YES
ADD 1 TO COUNT OF EQUATES
U9
U8A
FORM
SET UP RECORD IN NEW DICT. BUFFER
MOVE PRIMARY CODED VALUE TO WORD 2 OF NEW DICT
SAVEQ
GENERATE A DUMMY USER RECORD (EQUATE)
U10
SAVE=0 ?
NO
U11 A
YES
MOVE PRIMARY CODED VALUE TO WORD 2 OF NEW DICT.
SAVEQ
GENERATE A DUMMY USER RECORD (EQUATE)
ADD 1 TO COUNT OF EQUATES
U11 A

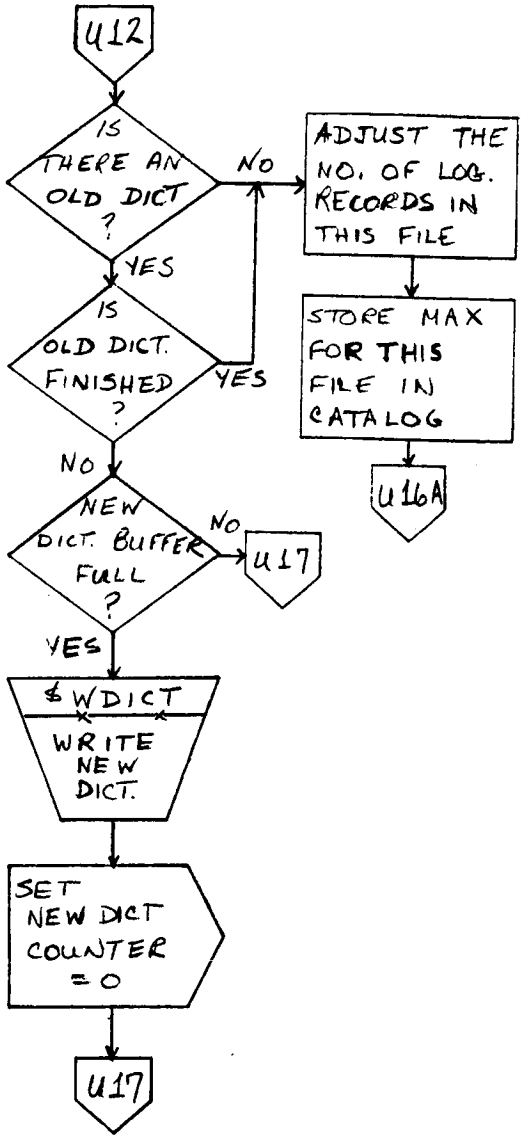
CR-62021

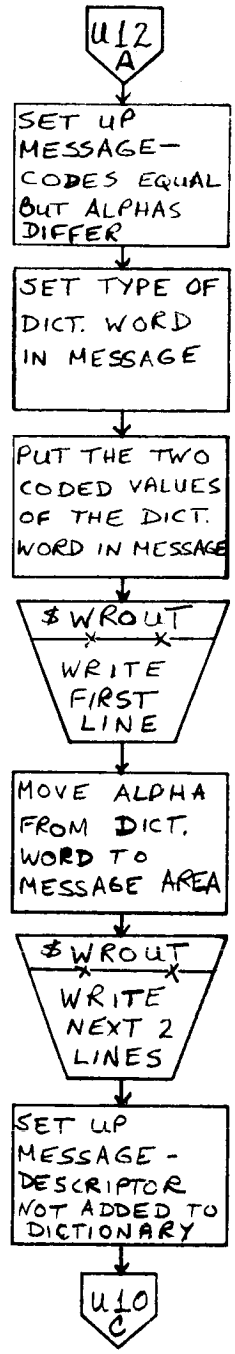

U10
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
SET UP MESSAGE - SECOND CODED VALUE CHANGED
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
SECOND CODE ALL BLANKS ?
NO
PUT SECOND CODED VALUE IN MESSAGE
YES
PUT BLANKS FOR SECONS CODE IN THE MESSAGE
$WROUT
WRITE FIRST LINE
COMM1
U10 B
SET TRA3=2 EXCEPT FOR DOC., USER, AND DICT DEL. CARDS
MAIN PHRASE ?
NO
TRA4= ?
YES
U9A
1
U11 A
2
U13 C
3
U7C
U10 A
SAVEQ
CANCEL The Dummy USER RECORD (EQUATE)
SET UP MESSAGE - DESCRIPTOR NOT A BASE
COMM
U10 C
STYPE
PUT TYPE OF DESCRIPTOR IN MESSAGE
PUT CODED VALUE OF DESCRIPTOR IN MESSAGE
PUT IN BLANKS FOR SECOND CODE IN THE MESSAGE
$WROUT
WRITE FIRST LINE
U10 B

CR 62021

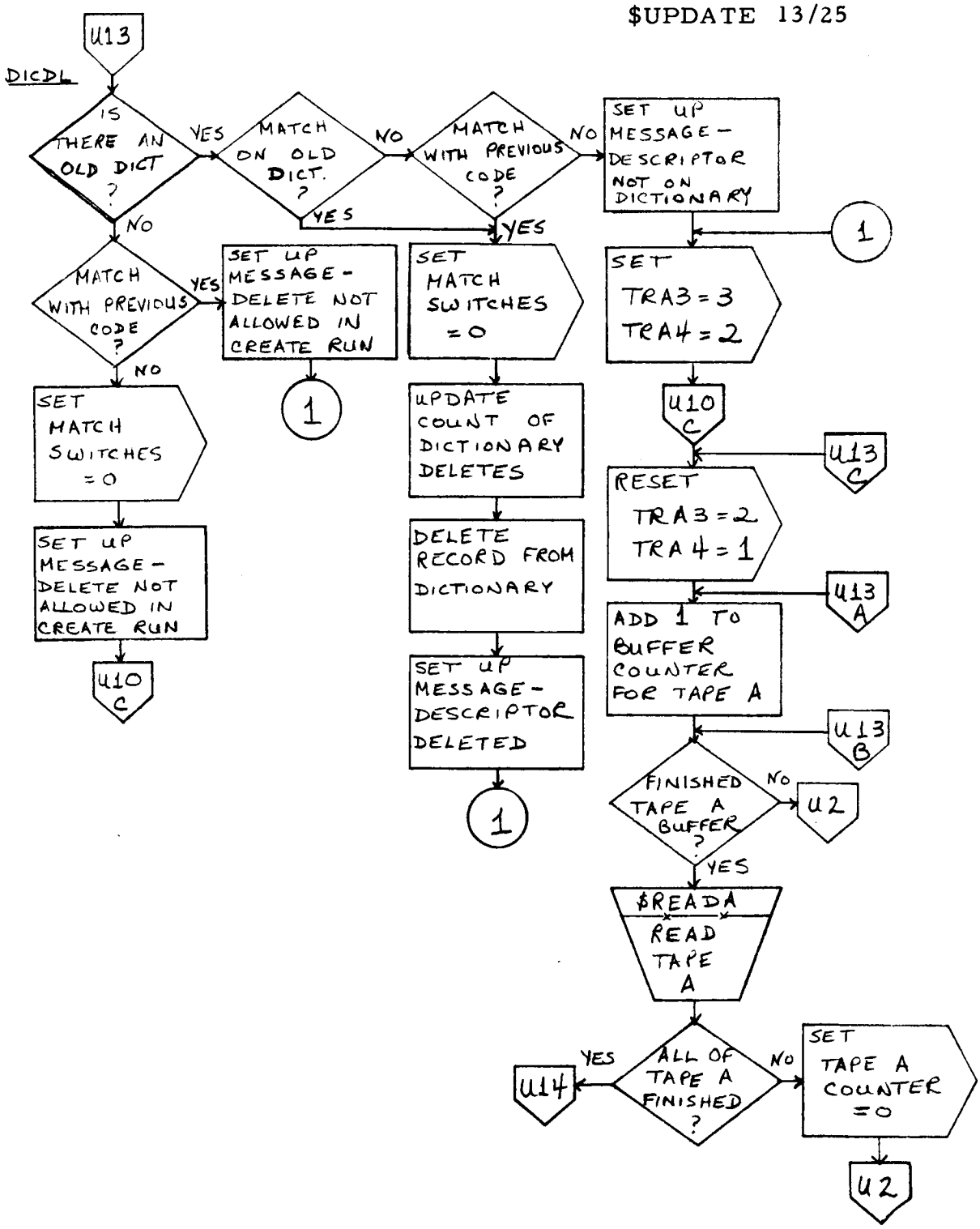

u11
DICAD
MATCH ON OLD DICT ?
NO
MATCH WITH PREVIOUS CODE ?
NO
YES
YES
ADD 1 TO COUNT OF DICTIONARY ADDS
FORM
SET UP RECORD IN NEW DICT BUFFER
SET UP MESSAGE - DESCRIPTOR ADDED TO DICTIONARY
u10 C
RESET MATCH SWITCHES
ALPHA ON DICT. BLANKS ?
YES
MOVE ALPHA FROM TAPE A TO MESSAGE AREA
SET UP MESSAGE - ONLY ALPHA CHANGED FOR DESCRIPTOR
NO
DO THE TWO ALPHMS MATCH ?
NO
u12 A
YES
SET UP MESSAGE - DESCRIPTOR ALREADY IN DICTIONARY
1
$WDICT
WRITE NEW DICT.
SET NEW DICT. COUNTER = 0
u2
u10 C
ANYA
u11 A
FINISHED TAPE A BUFFR ?
YES
$READA
READ TAPE A
ALL OF TAPE A FINISHED ?
YES
u11 B
NO
SET TAPE A COUNTER = 0
NO
NEXT CODE = PRESENT CODE ?
NO
u11 B
TRA 5 = 1 ?
NO
YES
u19
FINISHED A FILE OF NEW DICT ?
NO
YES
DONE
PROCESS THE END OF A NEW DICT FILE
NEW DICT. BUFFER FULL ?
YES
NO
ALL OF TAPE A FINISHED ?
YES
u12
NO
u2
YES
SET SWITCH INDICATING A MATCH WITH PREVIOUS CODE
u2
u2
1
NO
ALL OF TAPE A FINISHED ?
YES
u12
NO
u2

CR-62021

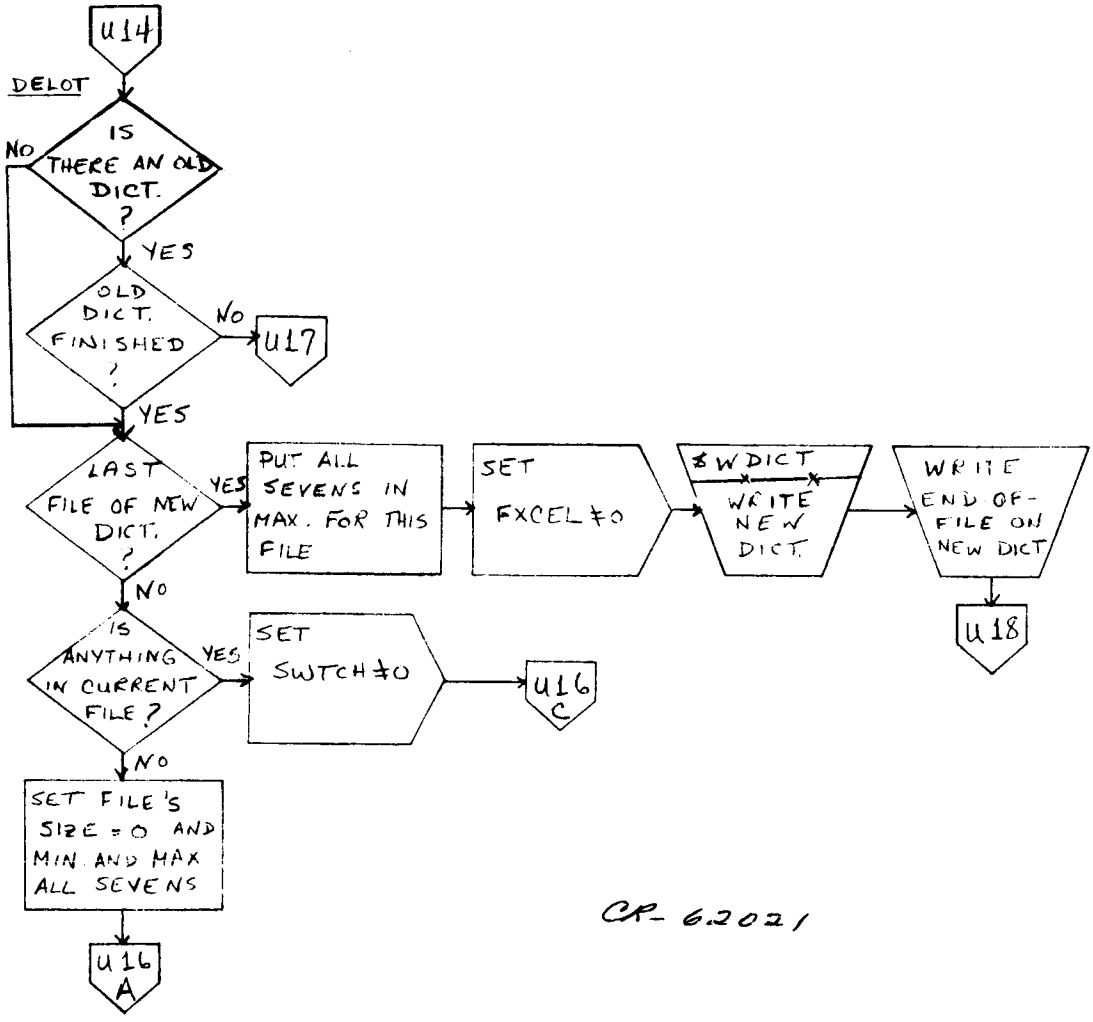

U12
IS THERE AN OLD DICT ?
NO
YES
ADJUST THE NO. OF LOG. RECORDS IN THIS FILE
STORE MAX FOR THIS FILE IN CATALOG
U16A
IS OLD DICT. FINISHED ?
YES
NO
NEW DICT. BUFFER FULL ?
NO
U17
YES
$WDICT
WRITE NEW DICT.
SET NEW DICT COUNTER = 0
U17


U12 A
SET UP MESSAGE - CODES EQUAL BUT ALPHAS DIFFER
SET TYPE OF DICT. WORD IN MESSAGE
PUT THE TWO CODED VALUES OF THE DICT. WORD IN MESSAGE
$WROUT
WRITE FIRST LINE
MOVE ALPHA FROM DICT. WORD TO MESSAGE AREA
$WROUT
WRITE NEXT 2 LINES
SET UP MESSAGE - DESCRIPTOR NOT ADDED TO DICTIONARY
U10 C

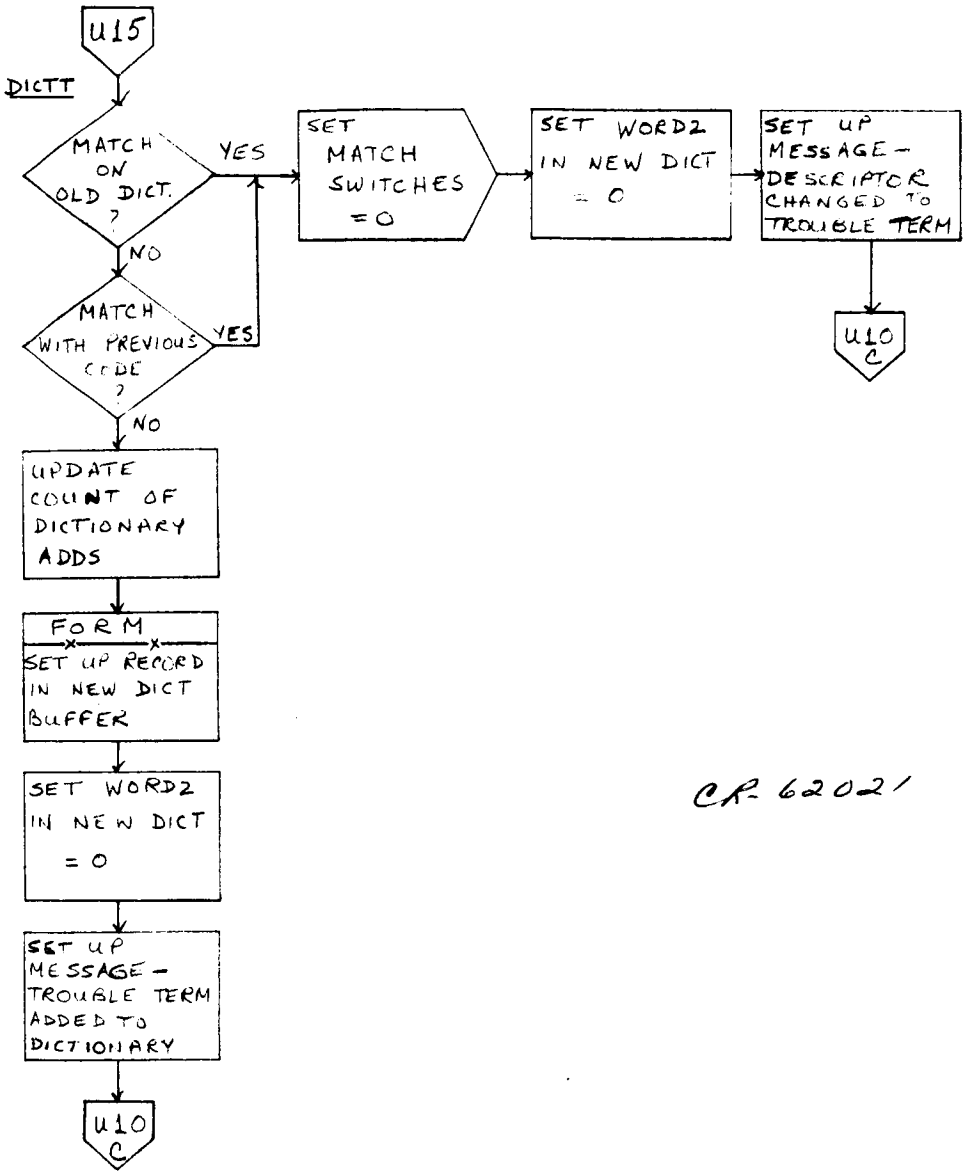
CR-62021


U13
DICDL
IS THERE AN OLD DICT ?
YES
MATCH ON OLD DICT. ?
NO
MATCH WITH PREVIOUS CODE ?
NO
SET UP MESSAGE - DESCRIPTOR NOT ON DICTIONARY
1
SET TRA3 = 3 TRA4 = 2
U10 C
U13 C
RESET TRA3 = 2 TRA4 = 1
U13 A
ADD 1 TO BUFFER COUNTER FOR TAPE A
U13 B
FINISHED TAPE A BUFFER ?
NO
U2
YES
$READA
READ TAPE A
ALL OF TAPE A FINISHED ?
YES
U14
NO
SET TAPE A COUNTER = 0
U2
YES
SET MATCH SWITCHES = 0
UPDATE COUNT OF DICTIONARY DELETES
DELETE RECORD FROM DICTIONARY
SET UP MESSAGE - DESCRIPTOR DELETED
1
MATCH WITH PREVIOUS CODE ?
YES
SET UP MESSAGE - DELETE NOT ALLOWED IN CREATE RUN
1
NO
SET MATCH SWITCHES = 0
SET UP MESSAGE - DELETE NOT ALLOWED IN CREATE RUN
U10 C

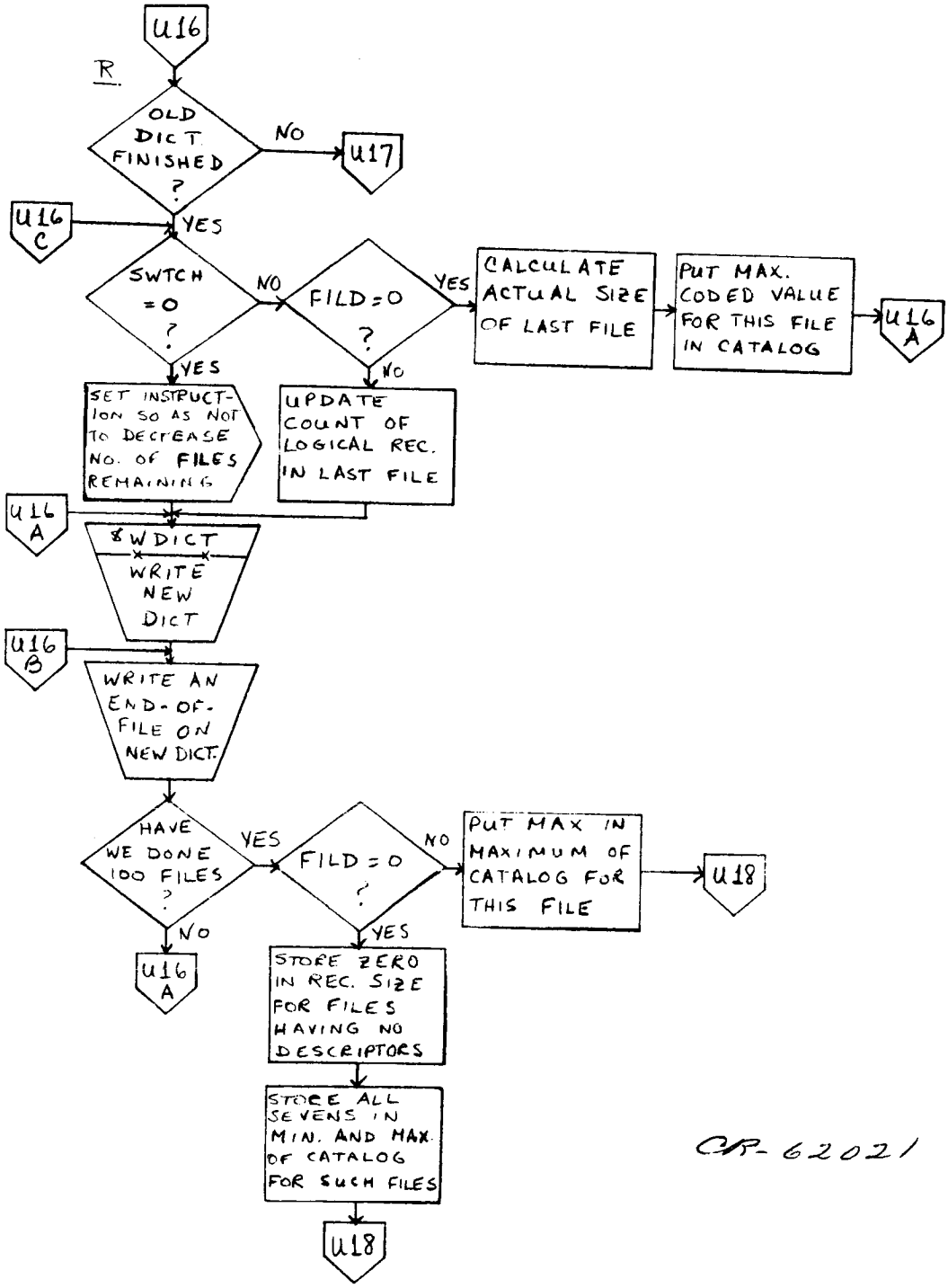
CR-62021


U14
DELOT
IS THERE AN OLD DICT. ?
NO
YES
OLD DICT. FINISHED ?
NO
U17
YES
LAST FILE OF NEW DICT. ?
YES
PUT ALL SEVENS IN MAX. FOR THIS FILE
SET FXCEL ≠ 0
$W DICT
WRITE NEW DICT.
WRITE END OF-FILE ON NEW DICT
U18
NO
IS ANYTHING IN CURRENT FILE ?
YES
SET SWTCH ≠ 0
U16 C
NO
SET FILE'S SIZE = 0 AND MIN AND MAX ALL SEVENS
U16 A

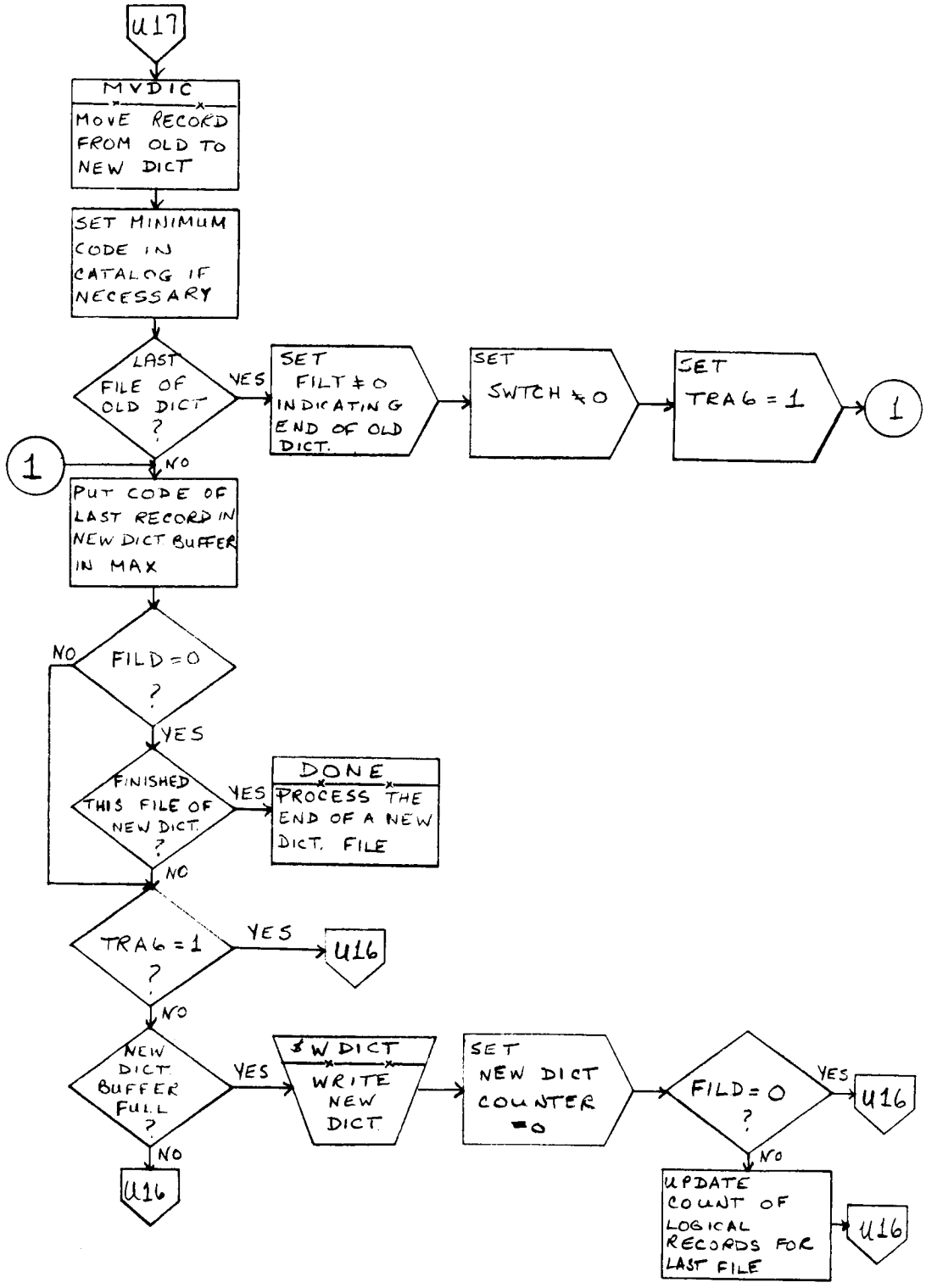
CR- 6.2021


U15
DICTT
MATCH ON OLD DICT. ?
YES
NO
SET MATCH SWITCHES = 0
SET WORD2 IN NEW DICT = 0
SET UP MESSAGE - DESCRIPTOR CHANGED TO TROUBLE TERM
U10 C
MATCH WITH PREVIOUS CODE ?
YES
NO
UPDATE COUNT OF DICTIONARY ADDS
FORM
SET UP RECORD IN NEW DICT BUFFER
SET WORD2 IN NEW DICT = 0
SET UP MESSAGE - TROUBLE TERM ADDED TO DICTIONARY
U10 C

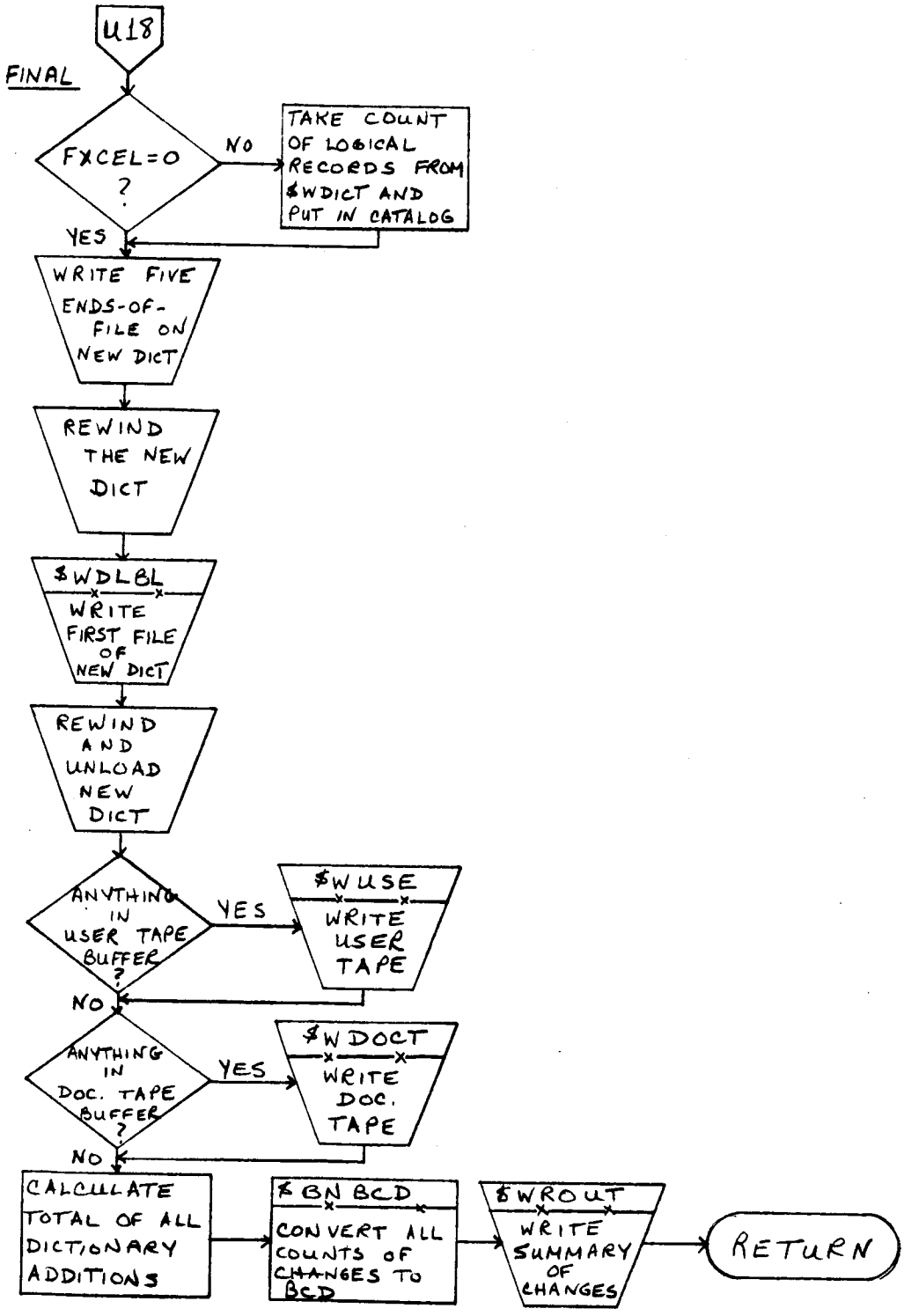
CR 62021


U16
R
OLD DICT FINISHED ?
NO
U17
U16 C
YES
SWTCH = 0 ?
NO
FILD=0 ?
YES
CALCULATE ACTUAL SIZE OF LAST FILE
PUT MAX. CODED VALUE FOR THIS FILE IN CATALOG
U16 A
YES
SET INSTRUCTION SO AS NOT TO DECREASE NO. OF FILES REMAINING
NO
UPDATE COUNT OF LOGICAL REC. IN LAST FILE
U16 A
$WDICT
WRITE NEW DICT
U16 B
WRITE AN END-OF-FILE ON NEW DICT.
HAVE WE DONE 100 FILES ?
YES
FILD = 0 ?
NO
PUT MAX IN MAXIMUM OF CATALOG FOR THIS FILE
U18
NO
U16 A
YES
STORE ZERO IN REC. SIZE FOR FILES HAVING NO DESCRIPTORS
STORE ALL SEVENS IN MIN. AND MAX. OF CATALOG FOR SUCH FILES
U18

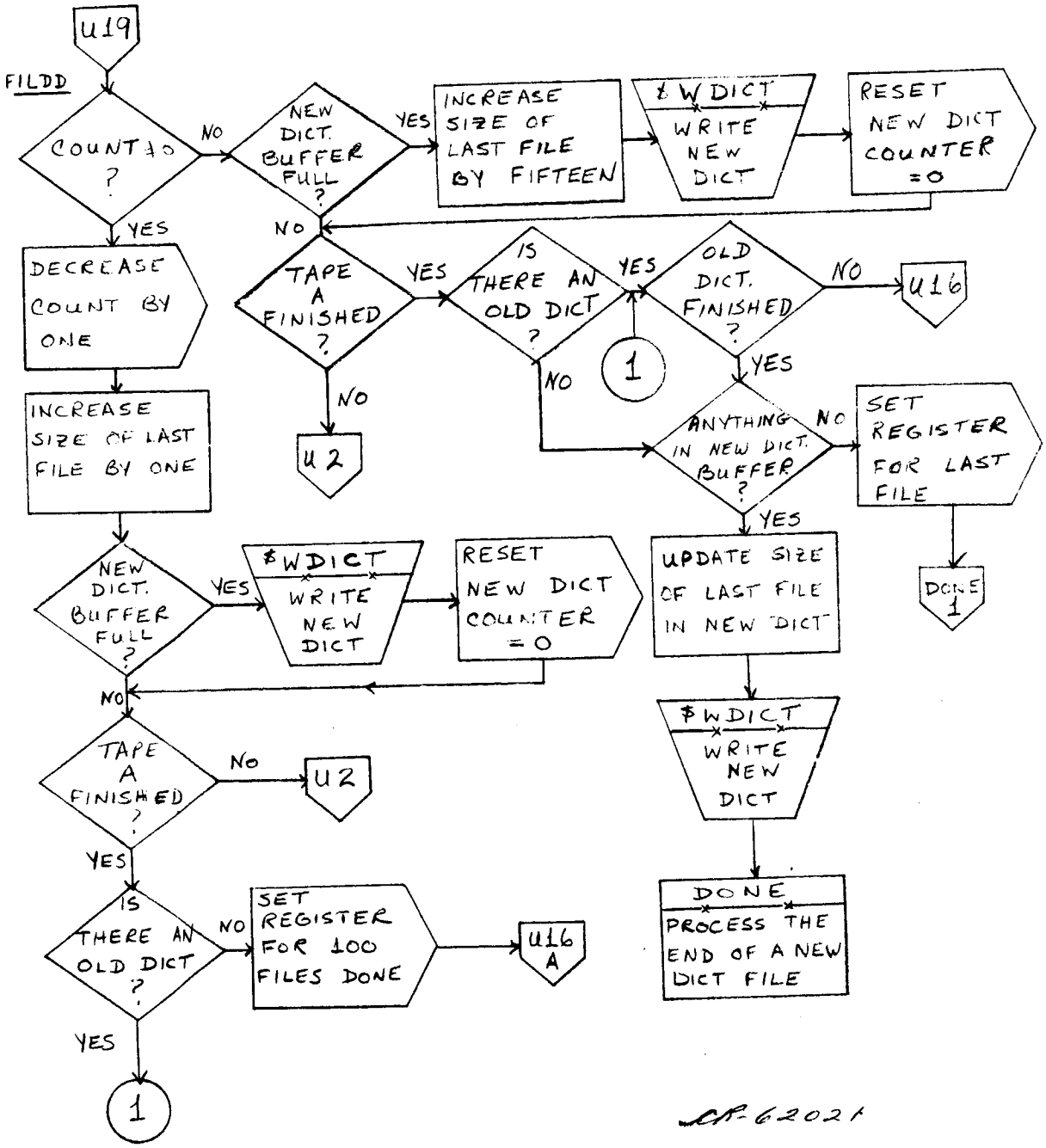
CR-62021

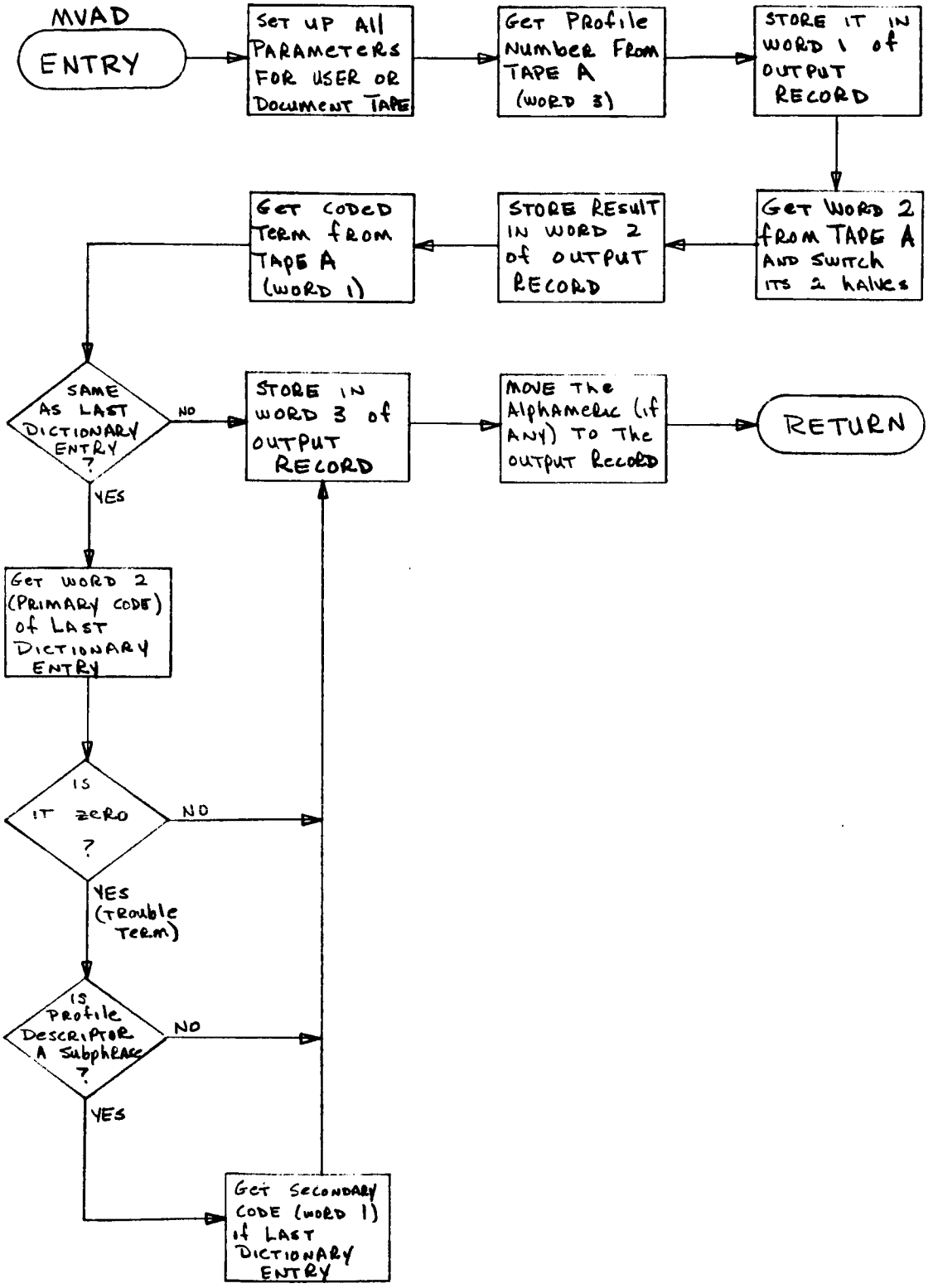

U17
MVDIC
MOVE RECORD FROM OLD TO NEW DICT
SET MINIMUM CODE IN CATALOG IF NECESSARY
LAST FILE OF OLD DICT ?
YES
SET FILT ≠ 0 INDICATING END OF OLD DICT.
SET SWTCH ≠ 0
SET TRA6 = 1
1
1
NO
PUT CODE OF LAST RECORD IN NEW DICT BUFFER IN MAX
NO
FILD = 0 ?
YES
FINISHED THIS FILE OF NEW DICT. ?
YES
DONE
PROCESS THE END OF A NEW DICT. FILE
NO
TRA6 = 1 ?
YES
U16
NO
NEW DICT. BUFFER FULL ?
YES
$W DICT
WRITE NEW DICT
SET NEW DICT COUNTER =0
FILD = 0 ?
YES
U16
NO
UPDATE COUNT OF LOGICAL RECORDS FOR LAST FILE
U16
NO
U16

CR-62021

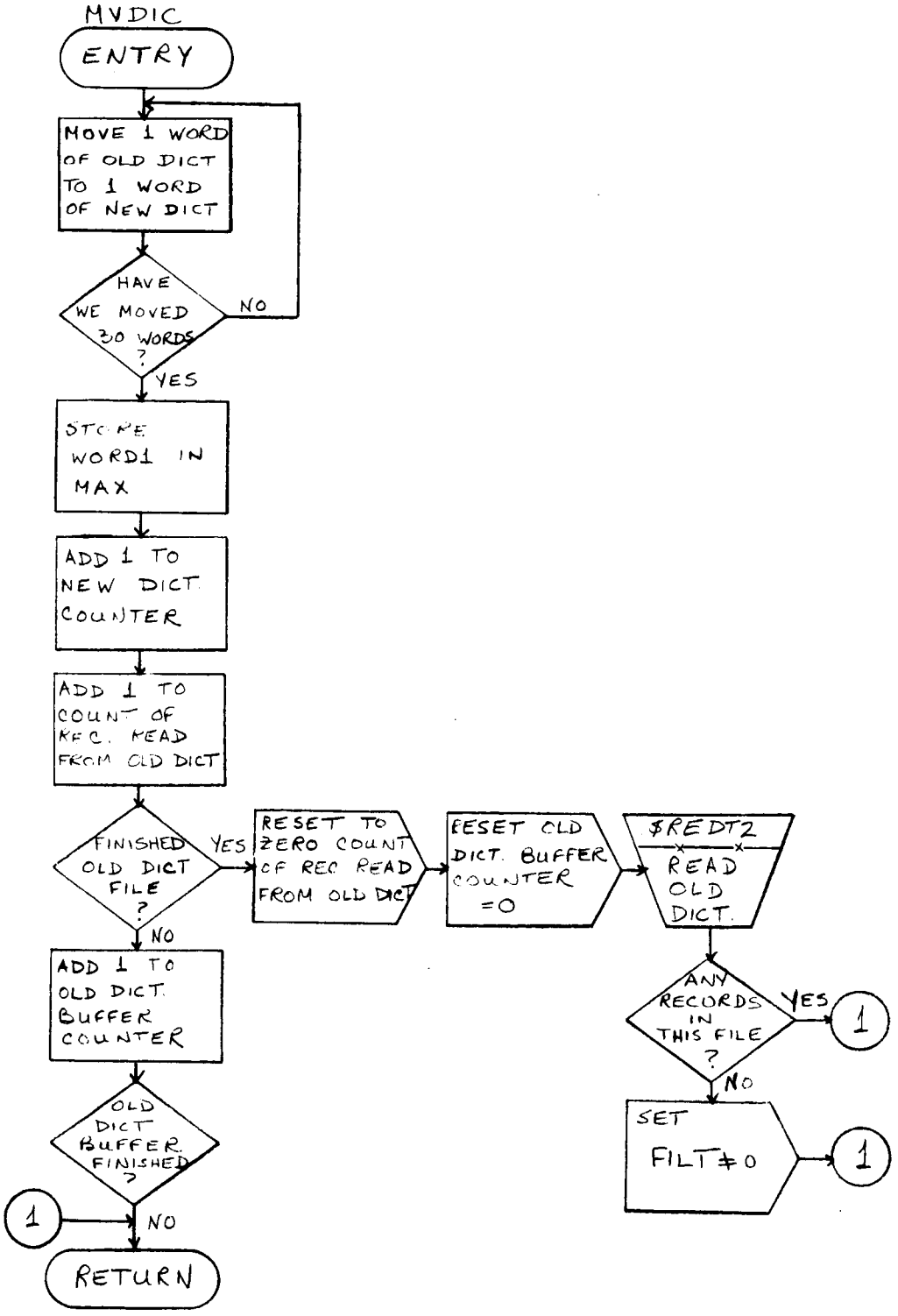

U18
FINAL
FXCEL=0 ?
NO
TAKE COUNT OF LOGICAL RECORDS FROM $WDICT AND PUT IN CATALOG
YES
WRITE FIVE ENDS-OF-FILE ON NEW DICT
REWIND THE NEW DICT
$WDLBL
WRITE FIRST FILE OF NEW DICT
REWIND AND UNLOAD NEW DICT
ANYTHING IN USER TAPE BUFFER ?
YES
$WUSE
WRITE USER TAPE
NO
ANYTHING IN DOC. TAPE BUFFER ?
YES
$WDOCT
WRITE DOC. TAPE
NO
CALCULATE TOTAL OF ALL DICTIONARY ADDITIONS
$BNBCD
CONVERT ALL COUNTS OF CHANGES TO BCD
$WROUT
WRITE SUMMARY OF CHANGES
RETURN

CR-62021

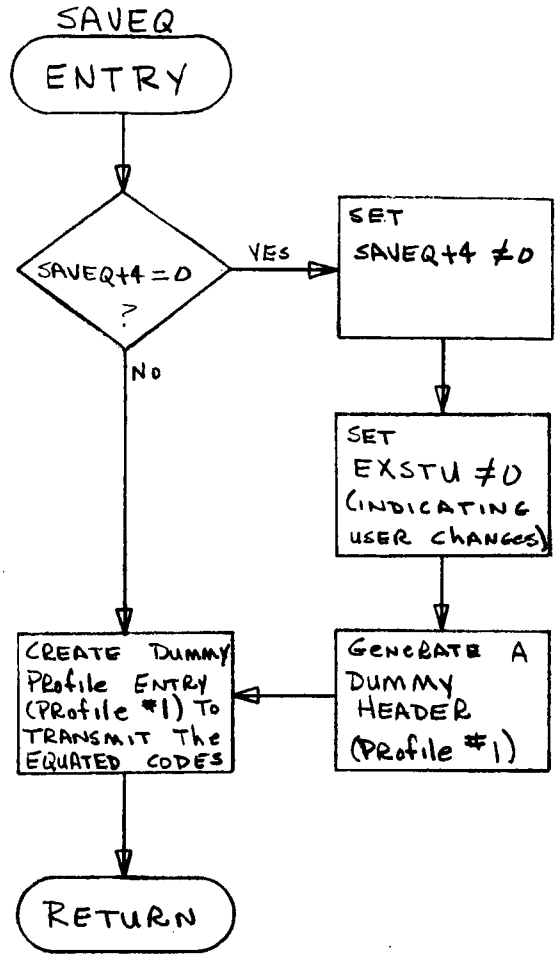

U19
FILDD
COUNT = 0 ?
NO
YES
NEW DICT. BUFFER FULL ?
YES
NO
INCREASE SIZE OF LAST FILE BY FIFTEEN
$W DICT
WRITE NEW DICT
RESET NEW DICT COUNTER = 0
DECREASE COUNT BY ONE
INCREASE SIZE OF LAST FILE BY ONE
TAPE A FINISHED ?
YES
NO
U2
IS THERE AN OLD DICT ?
YES
NO
1
OLD DICT. FINISHED ?
YES
NO
U16
ANYTHING IN NEW DICT. BUFFER ?
YES
NO
SET REGISTER FOR LAST FILE
DONE 1
UPDATE SIZE OF LAST FILE IN NEW DICT
$W DICT
WRITE NEW DICT
DONE
PROCESS THE END OF A NEW DICT FILE
NEW DICT. BUFFER FULL ?
YES
NO
$W DICT
WRITE NEW DICT
RESET NEW DICT COUNTER = 0
TAPE A FINISHED ?
NO
U2
YES
IS THERE AN OLD DICT ?
NO
YES
SET REGISTER FOR 100 FILES DONE
U16 A
1
CR-6202

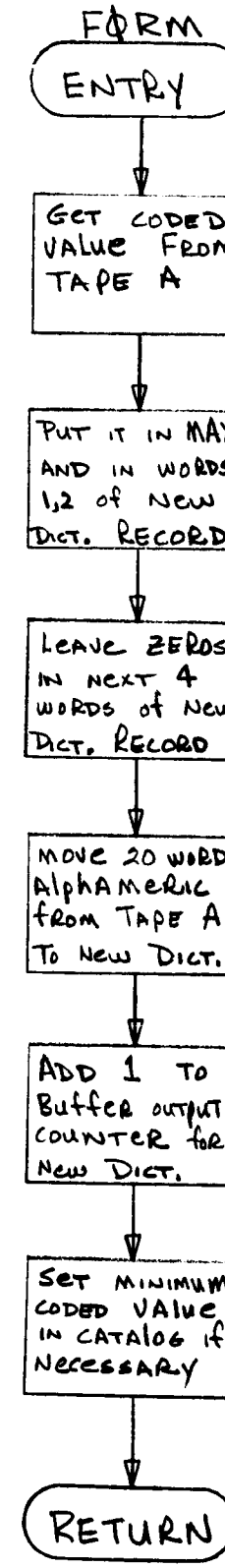

MVAD
ENTRY
Set up All Parameters for User or Document Tape
Get Profile Number From Tape A (word 3)
Store it in word 1 of output record
Get word 2 from Tape A and switch its 2 halves
Store result in word 2 of output record
Get coded term from Tape A (word 1)
Same as last Dictionary Entry ?
NO
YES
Store in word 3 of output record
Move the Alphameric (if any) to the output record
RETURN
Get word 2 (primary code) of last Dictionary Entry
Is it zero ?
NO
YES (Trouble Term)
Is Profile Descriptor A Subphrase ?
NO
YES
Get Secondary code (word 1) if last Dictionary Entry

CR 62021

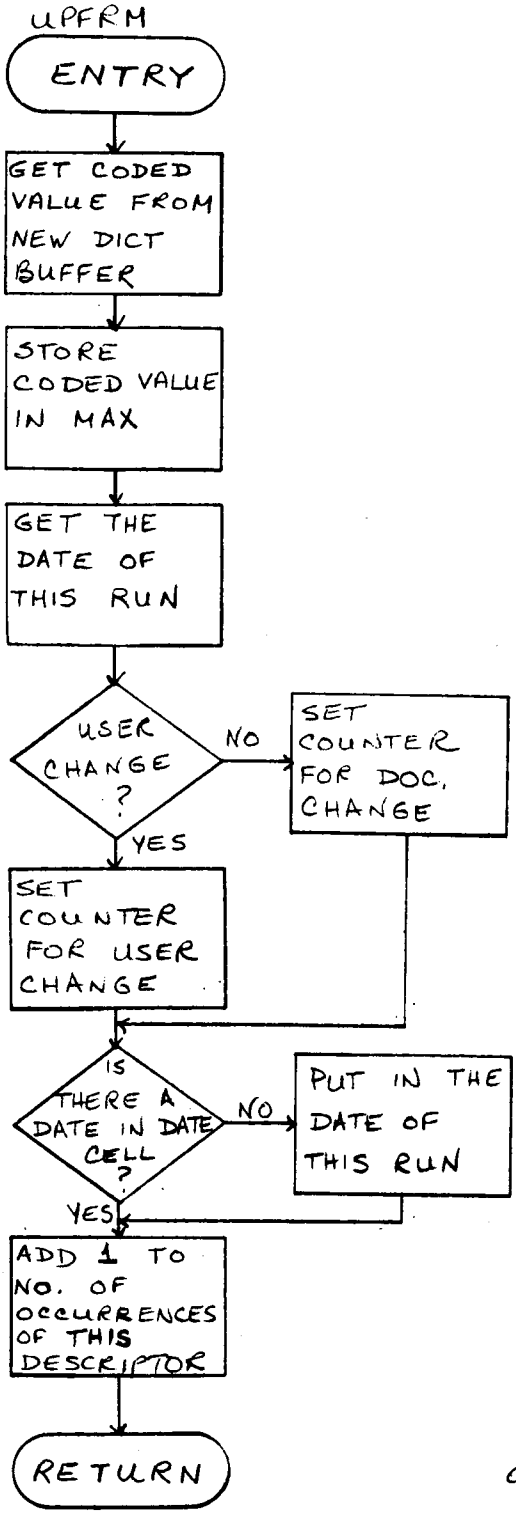

MVDIC
ENTRY
MOVE 1 WORD OF OLD DICT TO 1 WORD OF NEW DICT
HAVE WE MOVED 30 WORDS ?
NO
YES
STORE WORD1 IN MAX
ADD 1 TO NEW DICT COUNTER
ADD 1 TO COUNT OF REC. READ FROM OLD DICT
FINISHED OLD DICT FILE ?
YES
RESET TO ZERO COUNT OF REC READ FROM OLD DICT
RESET OLD DICT. BUFFER COUNTER =0
$REDT2
READ OLD DICT.
ANY RECORDS IN THIS FILE ?
YES
1
NO
SET FILT≠0
1
NO
ADD 1 TO OLD DICT. BUFFER COUNTER
OLD DICT BUFFER FINISHED ?
1
NO
RETURN

CR 62021

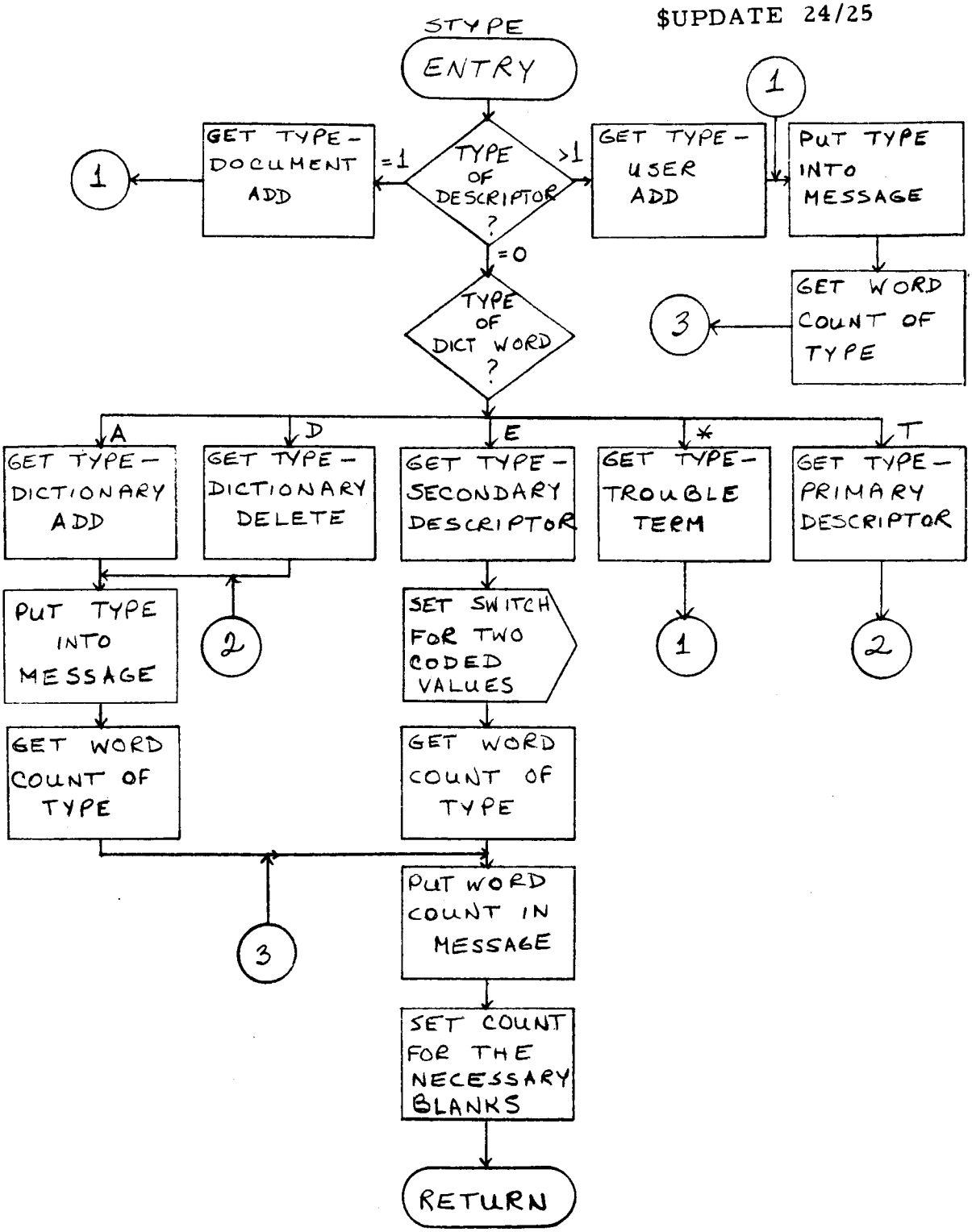

SAVEQ
ENTRY
SAVEQ+4 = 0 ?
YES
NO
SET SAVEQ+4 ≠0
SET EXSTU ≠0 (INDICATING USER CHANGES)
GENERATE A DUMMY HEADER (Profile #1)
CREATE DUMMY Profile ENTRY (Profile #1) TO TRANSMIT The EQUATED CODES
RETURN


FORM
ENTRY
GET CODED VALUE FROM TAPE A
PUT IT IN MAX AND IN WORDS 1,2 of NEW DICT. RECORD
LEAVE ZEROS IN NEXT 4 WORDS of NEW DICT. RECORD
MOVE 20 WORD AlphAMERIC from TAPE A To NEW DICT.
ADD 1 TO Buffer output COUNTER for NEW DICT.
SET MINIMUM CODED VALUE IN CATALOG if NECESSARY
RETURN

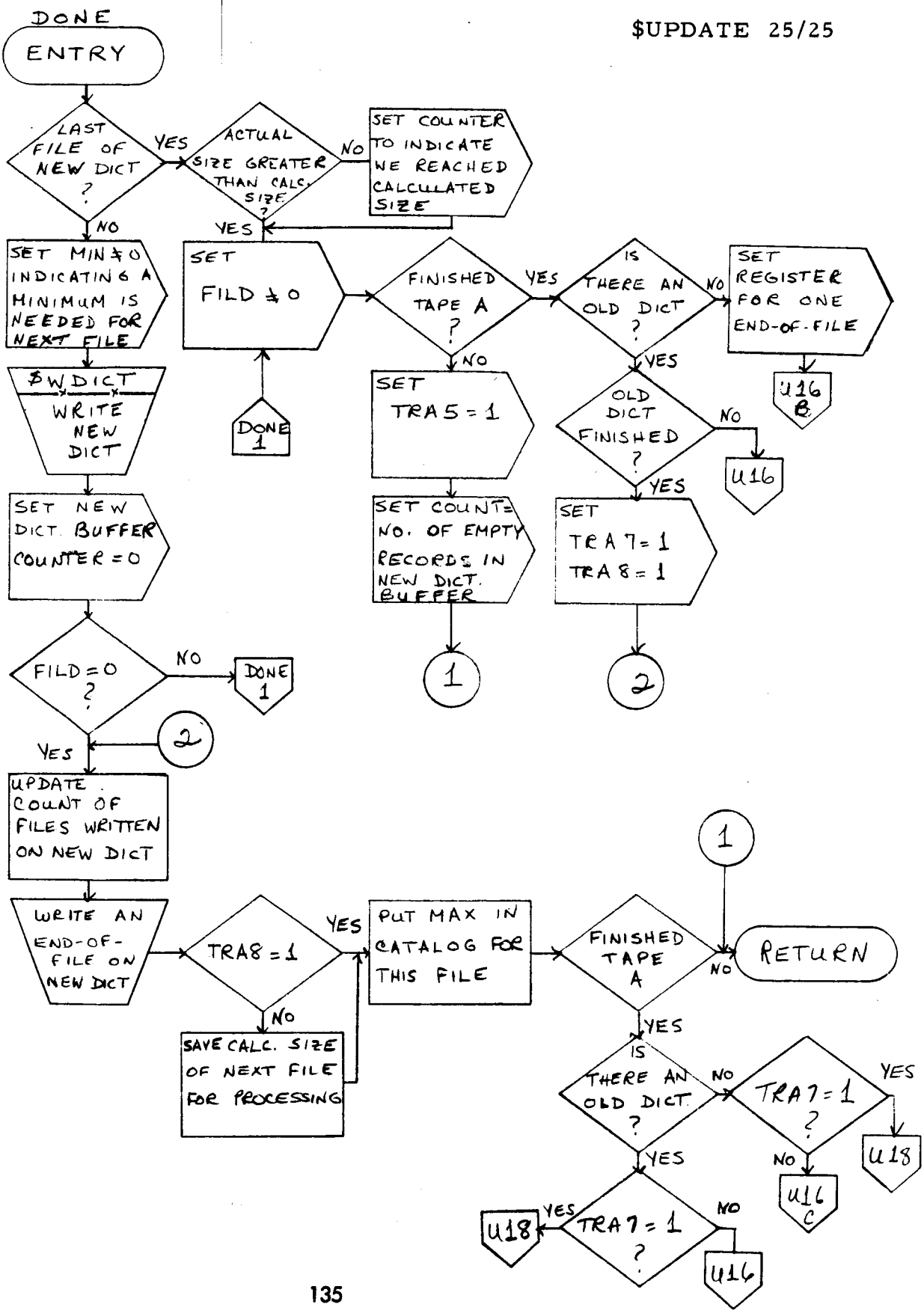
CR 62021


UPFRM
ENTRY
GET CODED VALUE FROM NEW DICT BUFFER
STORE CODED VALUE IN MAX
GET THE DATE OF THIS RUN
USER CHANGE ?
NO
SET COUNTER FOR DOC. CHANGE
YES
SET COUNTER FOR USER CHANGE
IS THERE A DATE IN DATE CELL ?
NO
PUT IN THE DATE OF THIS RUN
YES
ADD 1 TO NO. OF OCCURRENCES OF THIS DESCRIPTOR
RETURN

CR-62021


STYPE
ENTRY
TYPE OF DESCRIPTOR ?
=1
>1
=0
GET TYPE - DOCUMENT ADD
1
GET TYPE - USER ADD
1
PUT TYPE INTO MESSAGE
GET WORD COUNT OF TYPE
3
TYPE OF DICT WORD ?
A
D
E
*
T
GET TYPE - DICTIONARY ADD
GET TYPE - DICTIONARY DELETE
GET TYPE - SECONDARY DESCRIPTOR
GET TYPE - TROUBLE TERM
GET TYPE - PRIMARY DESCRIPTOR
PUT TYPE INTO MESSAGE
2
SET SWITCH FOR TWO CODED VALUES
1
2
GET WORD COUNT OF TYPE
GET WORD COUNT OF TYPE
3
PUT WORD COUNT IN MESSAGE
SET COUNT FOR THE NECESSARY BLANKS
RETURN

CR-6202'


DONE
ENTRY
LAST FILE OF NEW DICT ?
YES
ACTUAL SIZE GREATER THAN CALC. SIZE ?
NO
SET COUNTER TO INDICATE WE REACHED CALCULATED SIZE
NO
YES
SET MIN ≠ 0 INDICATING A MINIMUM IS NEEDED FOR NEXT FILE
SET FILD ≠ 0
FINISHED TAPE A ?
YES
IS THERE AN OLD DICT ?
NO
SET REGISTER FOR ONE END-OF-FILE
U16 B
$W DICT
WRITE NEW DICT
DONE 1
NO
SET TRA 5 = 1
YES
OLD DICT FINISHED ?
NO
U16
SET NEW DICT BUFFER COUNTER = 0
SET COUNT = NO. OF EMPTY RECORDS IN NEW DICT BUFFER
YES
SET TRA 7 = 1 TRA 8 = 1
FILD = 0 ?
NO
DONE 1
1
2
2
YES
UPDATE COUNT OF FILES WRITTEN ON NEW DICT
1
WRITE AN END-OF-FILE ON NEW DICT
TRA8 = 1
YES
PUT MAX IN CATALOG FOR THIS FILE
FINISHED TAPE A
NO
RETURN
NO
SAVE CALC. SIZE OF NEXT FILE FOR PROCESSING
YES
IS THERE AN OLD DICT ?
NO
TRA 7 = 1 ?
YES
U18
NO
U16 C
YES
TRA 7 = 1 ?
YES
U18
NO
U16

CR-62021

Subroutine UPDUS

The building and updating of the two user tapes (coded and historical) is done in this subroutine.

There are five sections within UPDUS that read or write all binary tapes. All are initialized upon their first entry and take appropriate action when encountering redundancy, EOF or EOT. These sections are:

READC	Read sorted tape C
USRID	Read old user tape
UHID	Read old historical profiles tape
WRUID	Write new user tape
WRHID	Write new historical profiles tape.

The FORTRAN II I/O Package is used whenever it is desired to write on the system output tape. This is accomplished with a TSX (TAPE), 4 and two calling sequence words.

First in UPDUS, all counters are set to zero. Flags are then set to show whether or not there is an IP user tape, IP historical profile tape and whether or not on an initial run a historical profile tape is to be generated.

If there is an IP user tape, its identification record is checked against the IP user control card. If there is not a match, a message will be printed on line and a dump will be taken. The same is done if there is an IP historical profiles tape.

The new identification records, with the date of the run, are written on the new user tape (and historical tape, if there is one).

The two types of runs, build and update, will be discussed separately.

Build. All input for building user tapes comes from tape C. The first descriptor for any user must be a header. If there is no header, the user will not be processed and a message will be written on the system output tape. Each add descriptor is checked to see if it should be added to the user profile. When all the adds have been processed, the percent that have been accepted will be compared to the required percentage (MINU) specified on one of the control cards. If the accept percentage is greater than or equal to MINU, the user profile will be written on the binary tape. If not, it will be rejected. If the profile has been accepted, it will be written on the historical profile tape if that tape is specified. Descriptor delete cards will not be processed on a build computer run.

Update. On an update run there are two, or possibly three, input tapes. The two required are tape C and the IP user tape. If this run is with historical profiles there will also be an IP historical profile tape.

CR 62021

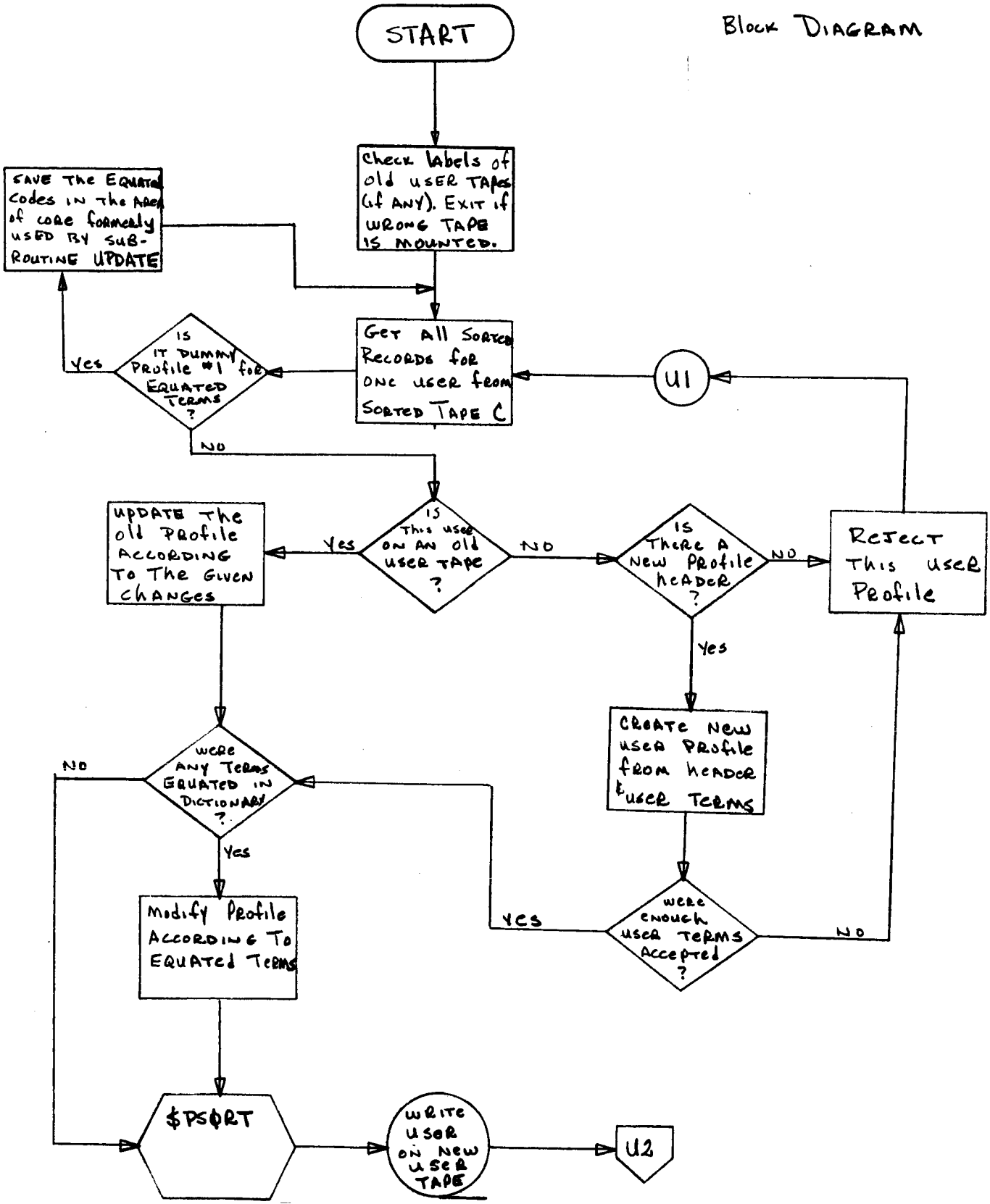
Users on the old tape(s) that do not appear on tape C are simply copied onto the new binary tape(s). When a user on the old tape also appears on tape C, the descriptors on tape C will be added to or deleted from the old user record. The appearance of a header card for an existing user will be interpreted as a new header and will be substituted for the original header. No comparison is made with MINU when updating an old user profile, but a new user, introduced in an update run, will be handled as in a build run.

Any changes to user profiles necessitated by vocabulary equate changes (establishing synonyms) were transmitted earlier by subroutine UPDATE in a dummy user profile (profile #1). These are gathered and stored in the area of core previously used by subroutine UPDATE. Each profile written onto the new user tape is compared with the newly created synonym relationships and is modified as required.

CR. 62021

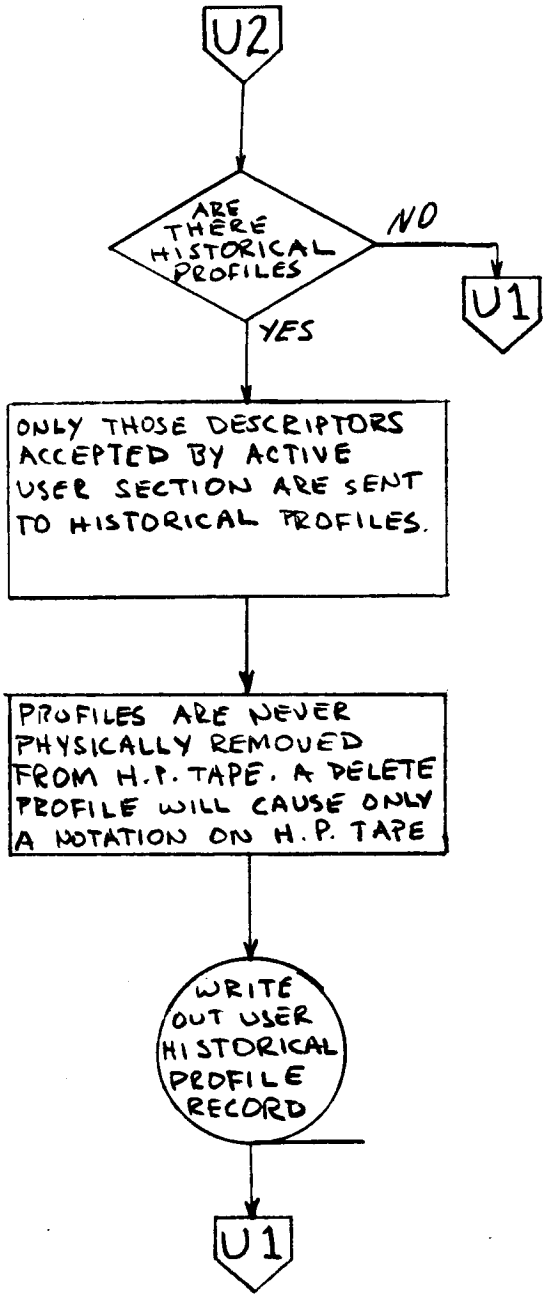
$UPDUS 1/2

Block Diagram



CR-62021

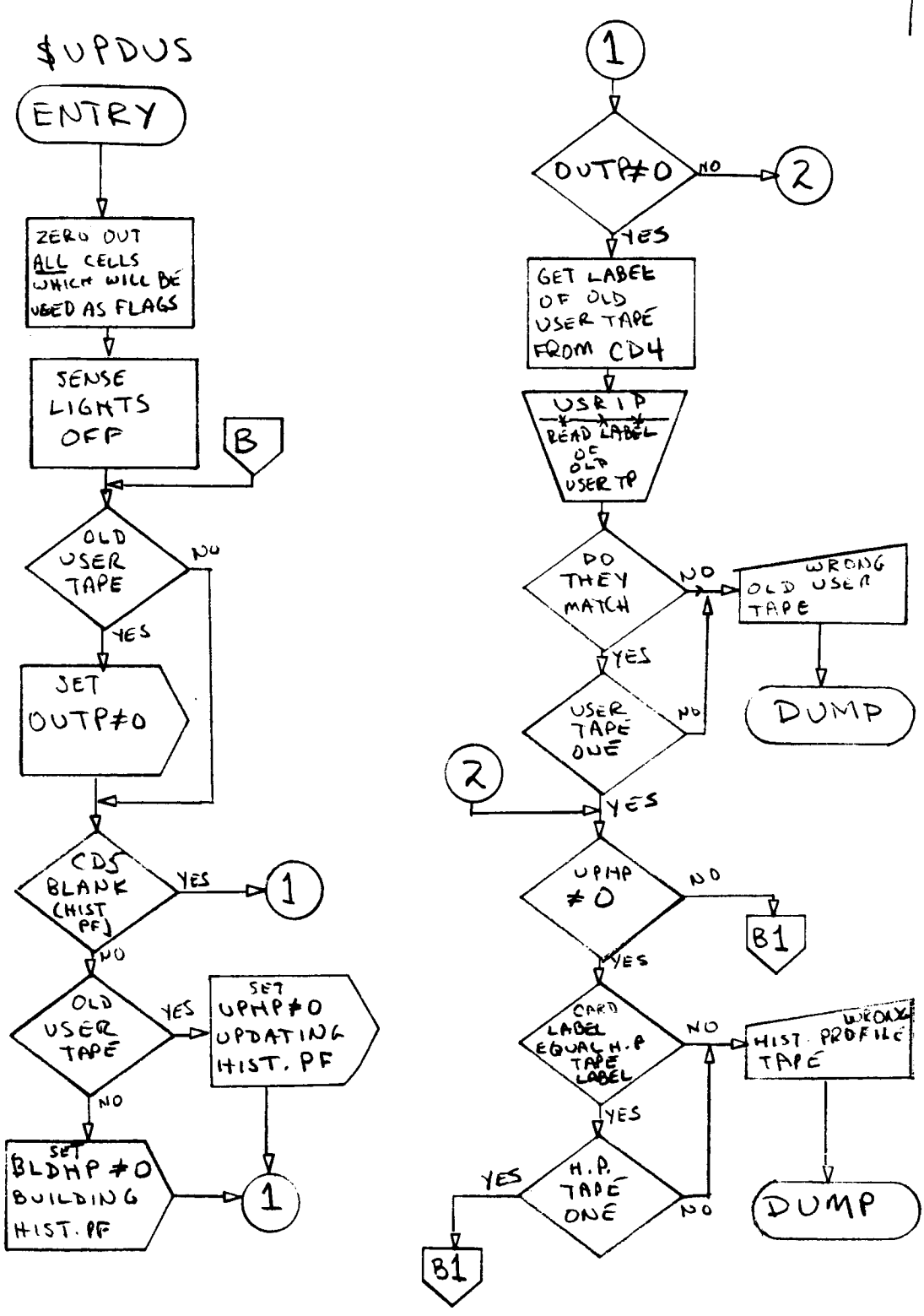
Block Diagram



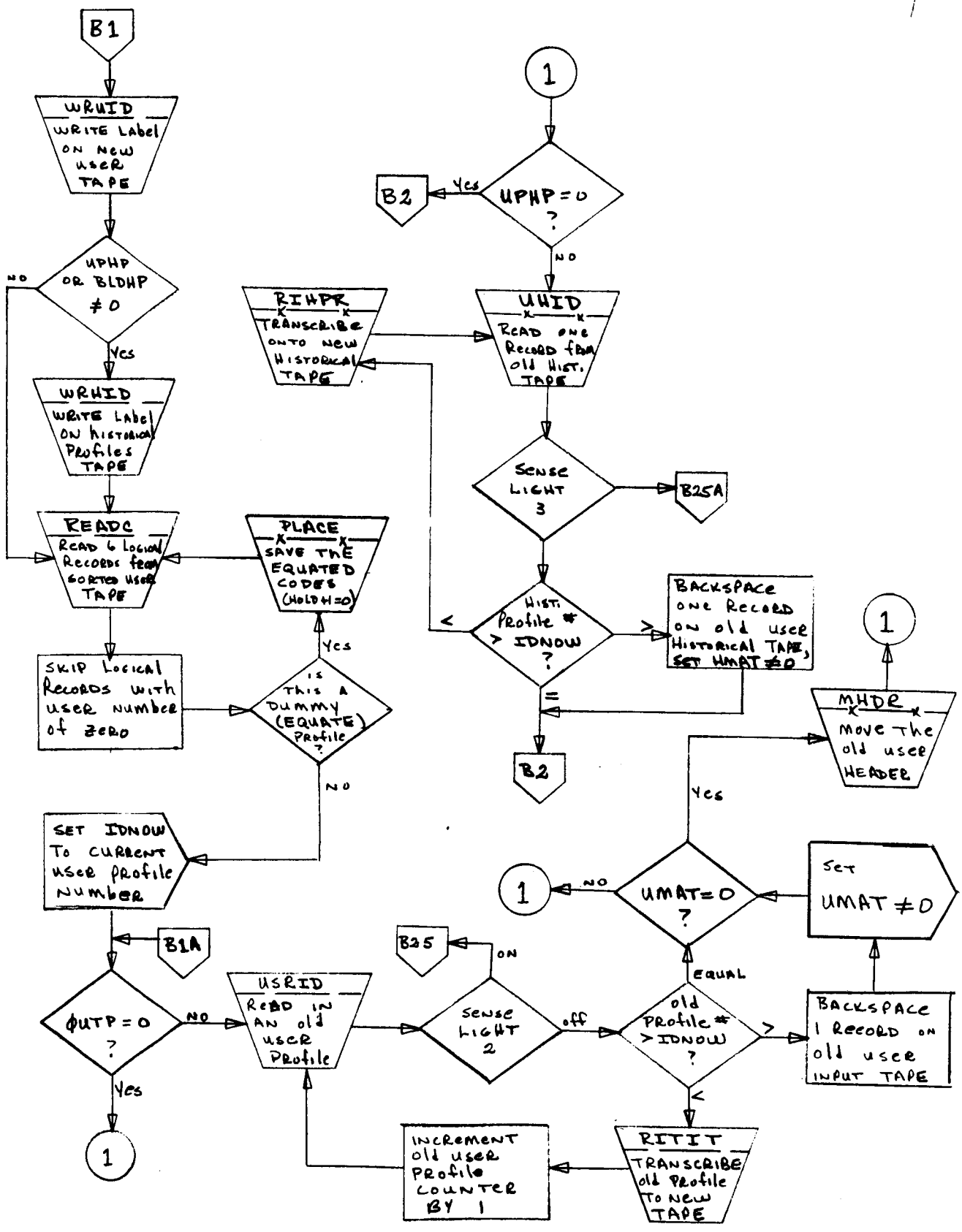
EVERY DESCRIPTOR, WHETHER ACCEPTED OR REJECTED IS SO NOTED ON THE BCD OUTPUT TAPE. THE SAME IS DONE FOR EACH PROFILE.

ON AN UPDATE RUN, ALL USERS WHO ARE NOT UPDATED ARE SIMPLY COPIED FROM OLD TAPE TO NEW TAPE.

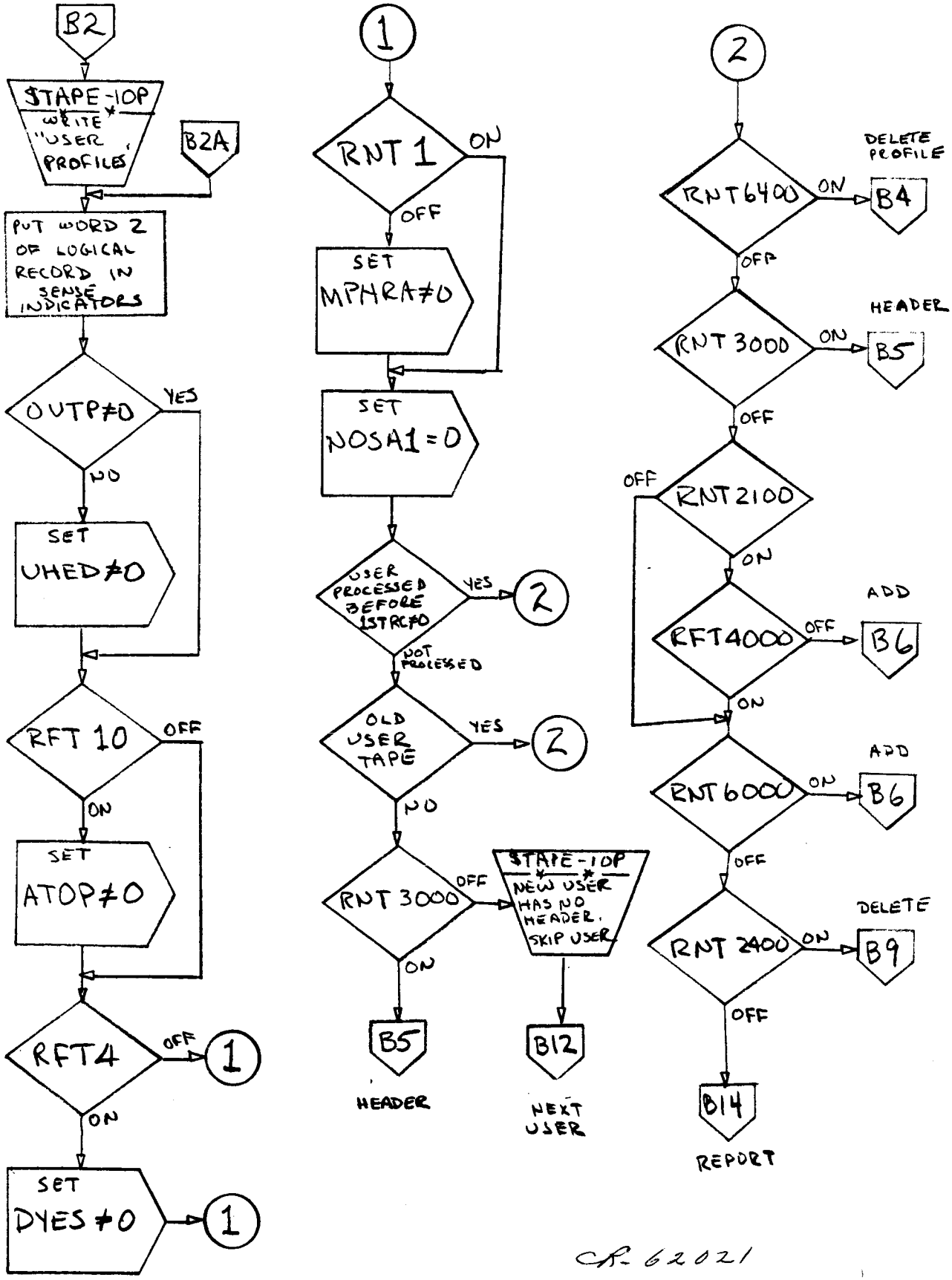
AR 62021


$UPDUS
ENTRY
ZERO OUT ALL CELLS WHICH WILL BE USED AS FLAGS
SENSE LIGHTS OFF
B
OLD USER TAPE
NO
YES
SET OUTP≠0
CD5 BLANK (HIST PF)
YES
1
NO
OLD USER TAPE
YES
SET UPHP≠0 UPDATING HIST. PF
NO
SET BLDHP≠0 BUILDING HIST. PF
1
1
OUTP≠0
NO
2
YES
GET LABEL OF OLD USER TAPE FROM CD4
USR I P
READ LABEL OF OLD USER TP
DO THEY MATCH
NO
WRONG OLD USER TAPE
DUMP
YES
USER TAPE ONE
NO
2
YES
UPHP ≠ 0
NO
B1
YES
CARD LABEL EQUAL H.P TAPE LABEL
NO
WRONG HIST PROFILE TAPE
DUMP
YES
H.P. TAPE ONE
NO
YES
B1

CR-62021

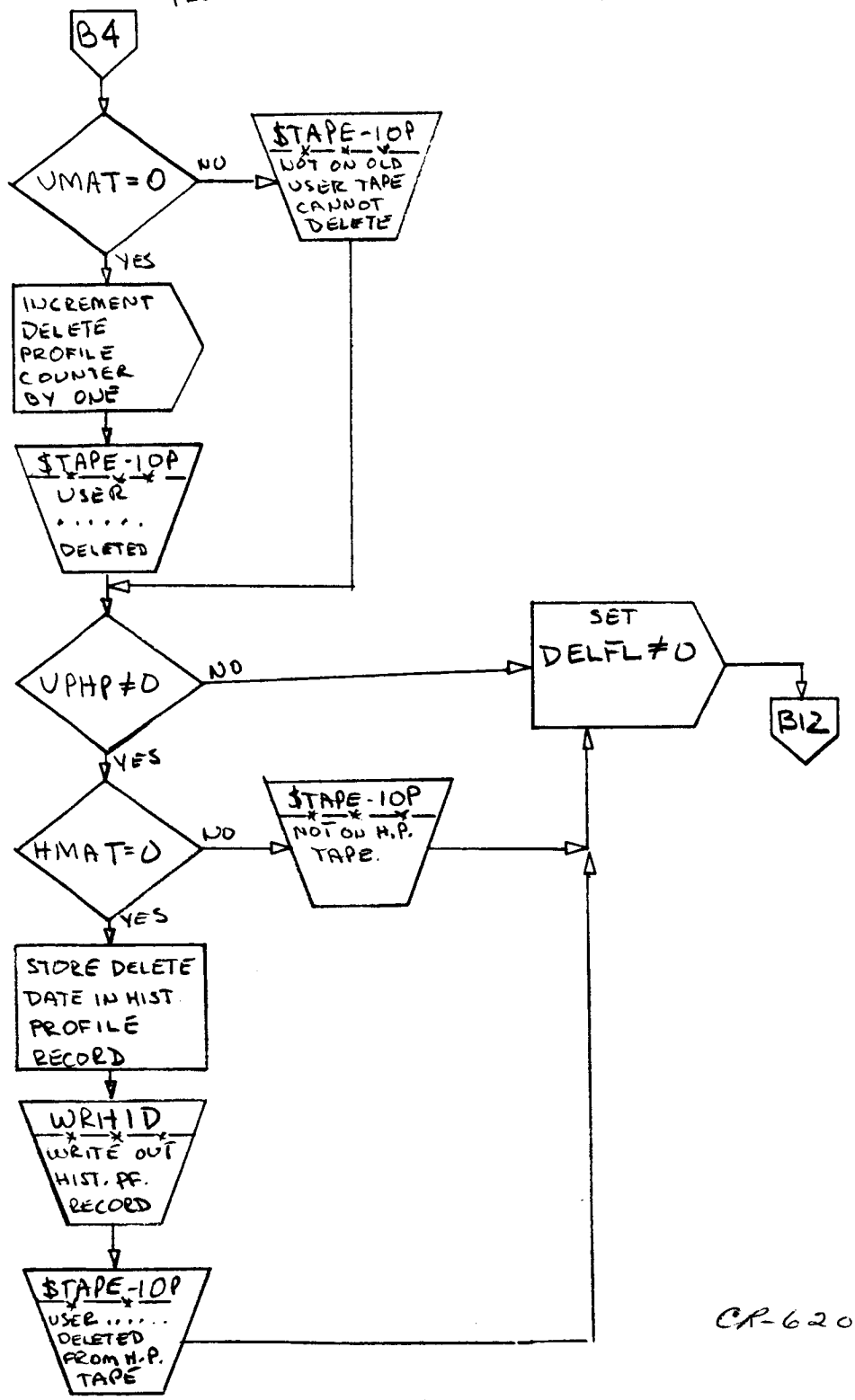

B1
WRUID
WRITE Label ON NEW USER TAPE
UPHP OR BLDHP ≠ 0
NO
Yes
WRHID
WRITE Label ON historical Profiles TAPE
READC
Read 6 Logical Records from SORTED USER TAPE
SKIP Logical Records with user number of zero
Is this a DUMMY (EQUATE) Profile ?
Yes
PLACE
SAVE THE EQUATED CODES (HOLD H=0)
NO
SET IDNOW To CURRENT user profile NUMBER
B1A
ϕUTP = 0 ?
NO
Yes
1
USRID
Read IN AN old USER Profile
B25
ON
Sense LIGHT 2
off
INCREMENT old user Profile COUNTER BY 1
RITIT
TRANSCRIBE old Profile To NEW TAPE
old Profile # > IDNOW ?
EQUAL
>
BACKSPACE 1 RECORD ON old user INPUT TAPE
<
UMAT=0 ?
NO
1
Yes
SET UMAT ≠ 0
MHDR
Move the old user HEADER
1
B2
Yes
UPHP = 0 ?
NO
1
RIHPR
TRANSCRIBE ONTO NEW HISTORICAL TAPE
UHID
Read one Record from old Hist. TAPE
Sense LIGHT 3
B25A
HIST. Profile # > IDNOW ?
<
>
BACKSPACE one Record on old user Historical TAPE, SET UMAT ≠ 0
=
B2

CR-62021


B2
$TAPE-10P
* WRITE "USER PROFILES"
B2A
PUT WORD 2 OF LOGICAL RECORD IN SENSE INDICATORS
OUTP≠0
YES
NO
SET UHED≠0
RFT 10
OFF
ON
SET ATOP≠0
RFT4
OFF
1
ON
SET DYES≠0
1
RNT 1
ON
OFF
SET MPHRA≠0
SET NOSA1=0
USER PROCESSED BEFORE ISTRC≠0
YES
2
NOT PROCESSED
OLD USER TAPE
YES
2
NO
RNT 3000
OFF
$TAPE-10P
* NEW USER HAS NO HEADER, SKIP USER
ON
B5
HEADER
B12
NEXT USER
2
RNT6400
ON
B4
DELETE PROFILE
OFF
RNT 3000
ON
B5
HEADER
OFF
RNT 2100
OFF
ON
RFT4000
OFF
B6
ADD
ON
RNT 6000
ON
B6
ADD
OFF
RNT 2400
ON
B9
DELETE
OFF
B14
REPORT

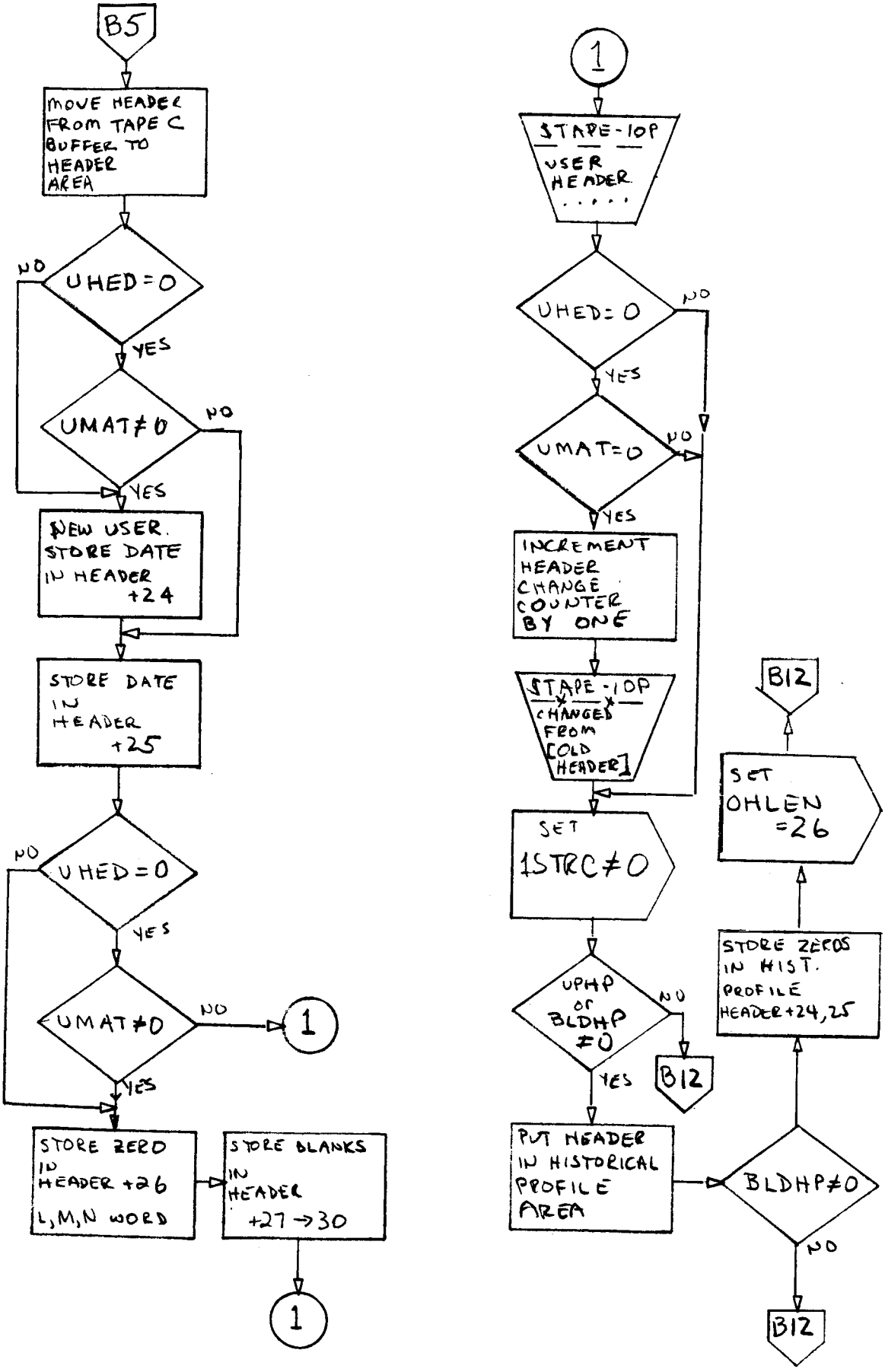
CR-62021

DELETE PROFILE

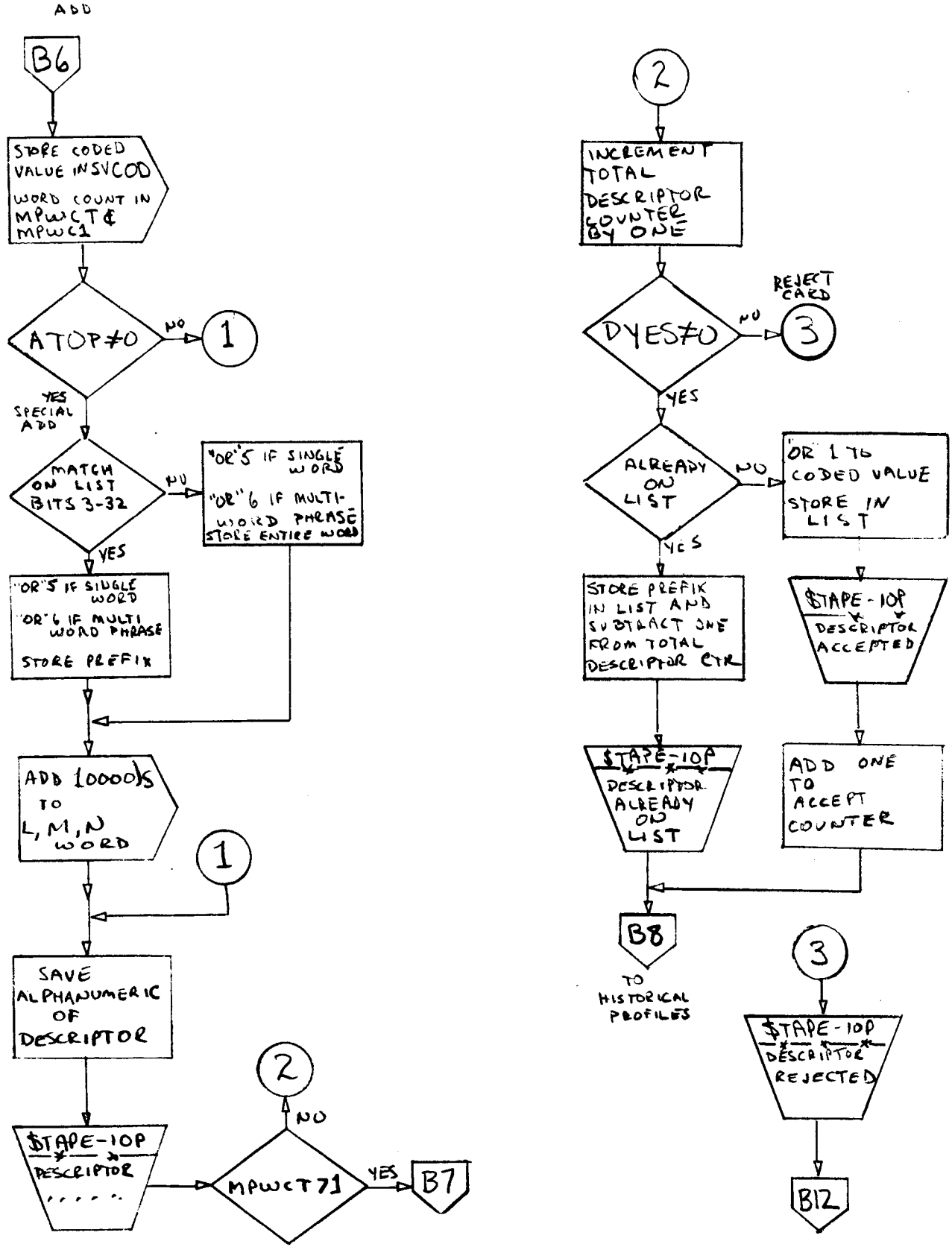


CR-62021

HEADER

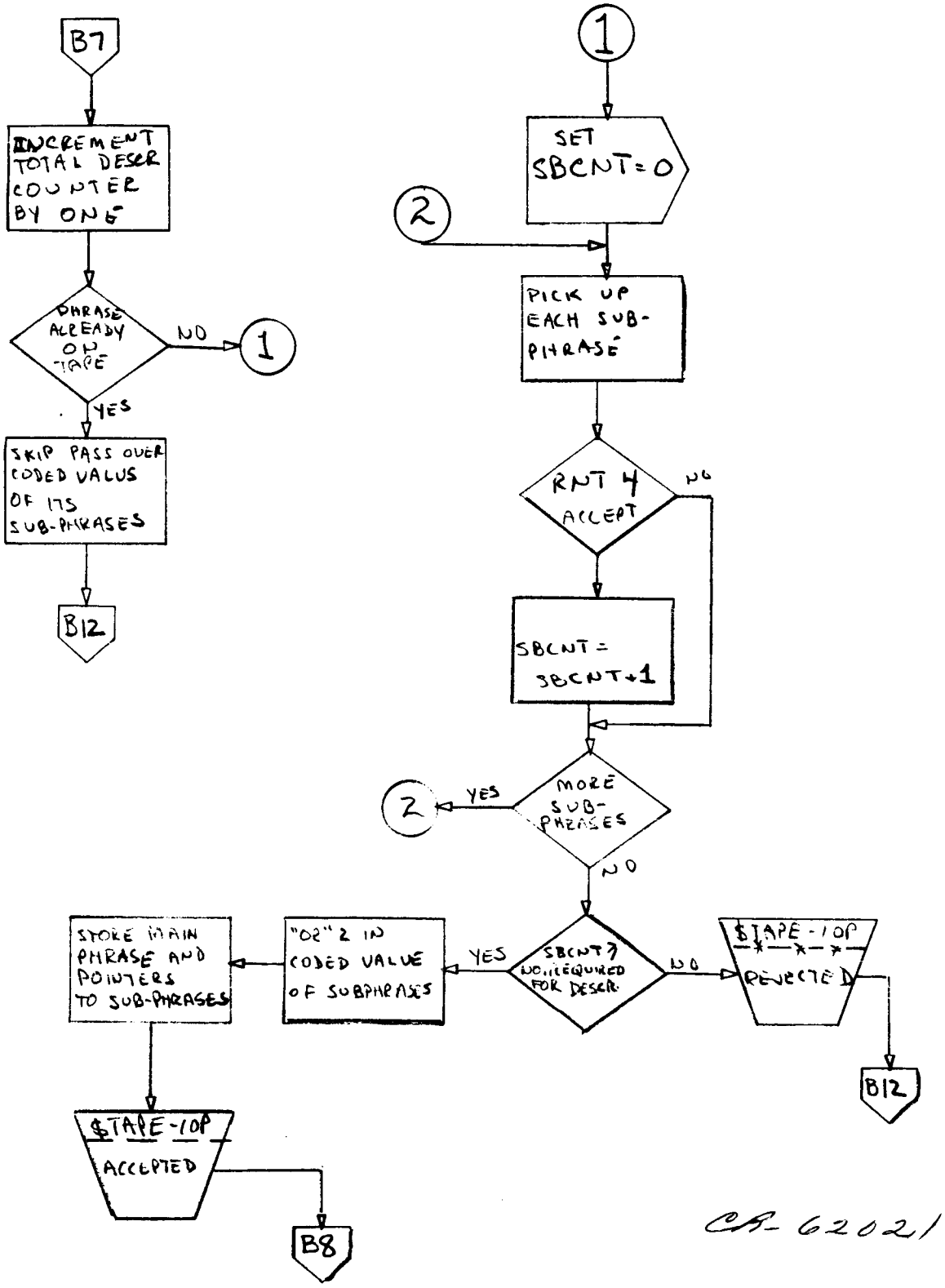


CR-62021

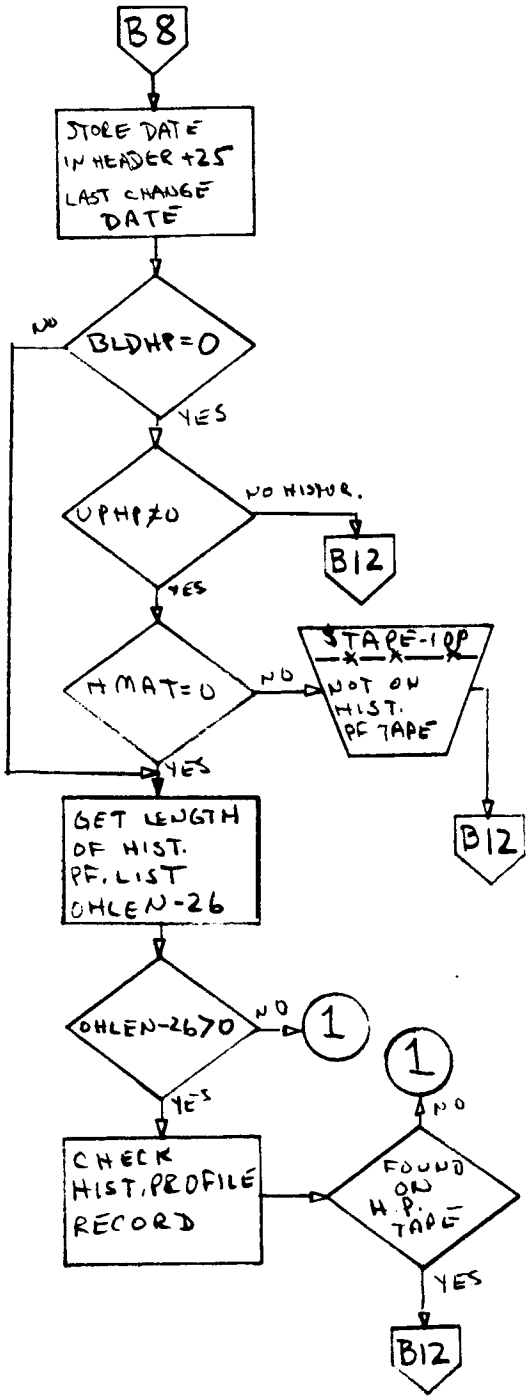


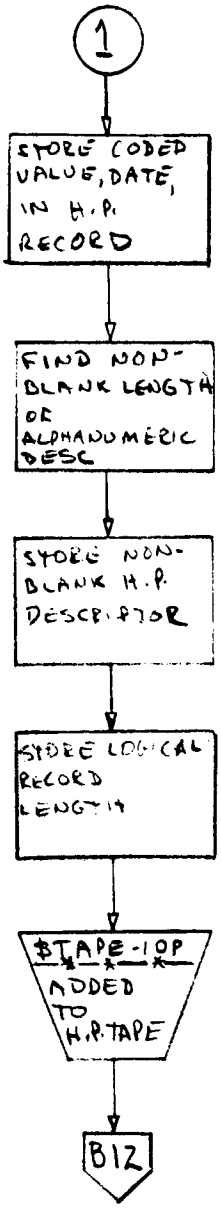
CR-62021

1 blank

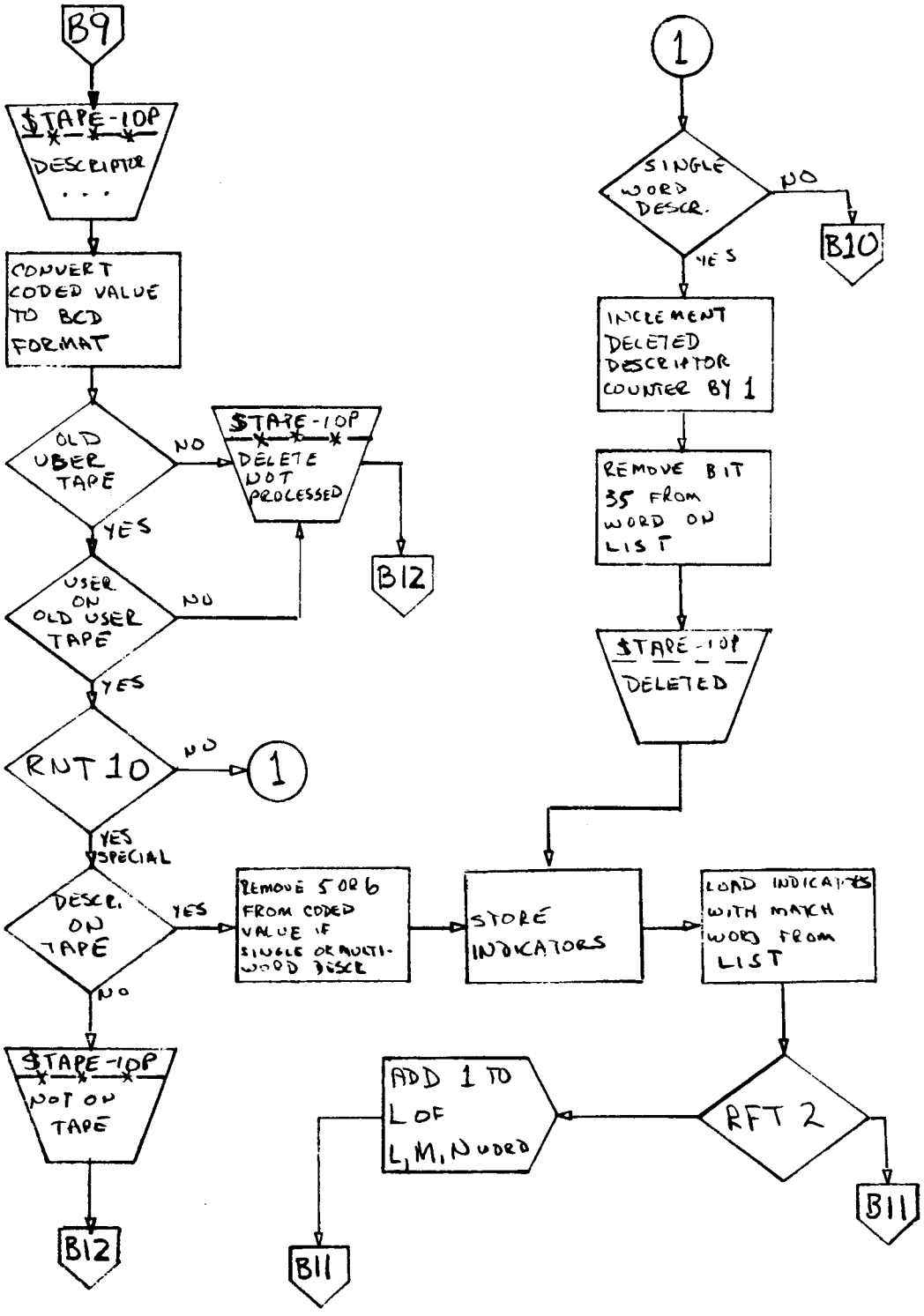

B7
INCREMENT TOTAL DESCR COUNTER BY ONE
PHRASE ALREADY ON TAPE
NO
1
YES
SKIP PASS OVER CODED VALUS OF ITS SUB-PHRASES
B12
1
SET SBCNT=0
2
PICK UP EACH SUB-PHRASE
RNT 4 ACCEPT
NO
SBCNT = SBCNT+1
2
YES
MORE SUB-PHRASES
NO
STORE MAIN PHRASE AND POINTERS TO SUB-PHRASES
"02"2 IN CODED VALUE OF SUBPHRASES
YES
SBCNT? NO. REQUIRED FOR DESCR
NO
$TAPE-IOP REJECTED
B12
$TAPE-IOP ACCEPTED
B8

CR-62021

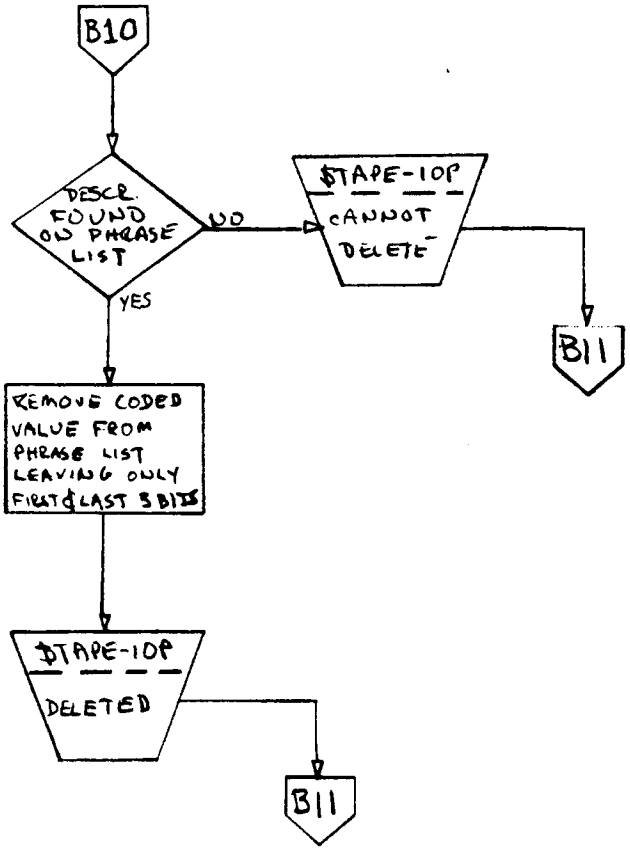

B8
STORE DATE IN HEADER +25 LAST CHANGE DATE
BLDHP=0
NO
YES
UPHP≠0
NO HISTOR.
B12
YES
HMAT=0
NO
$TAPE-IOP
NOT ON HIST. PF TAPE
B12
YES
GET LENGTH OF HIST. PF. LIST OHLEN-26
OHLEN-2670
NO
1
YES
CHECK HIST. PROFILE RECORD
FOUND ON H.P. TAPE
NO
1
YES
B12


1
STORE CODED VALUE, DATE, IN H.P. RECORD
FIND NON-BLANK LENGTH OF ALPHANUMERIC DESC
STORE NON-BLANK H.P. DESCRIPTOR
STORE LOGICAL RECORD LENGTH
$TAPE-IOP
ADDED TO H.P. TAPE
B12

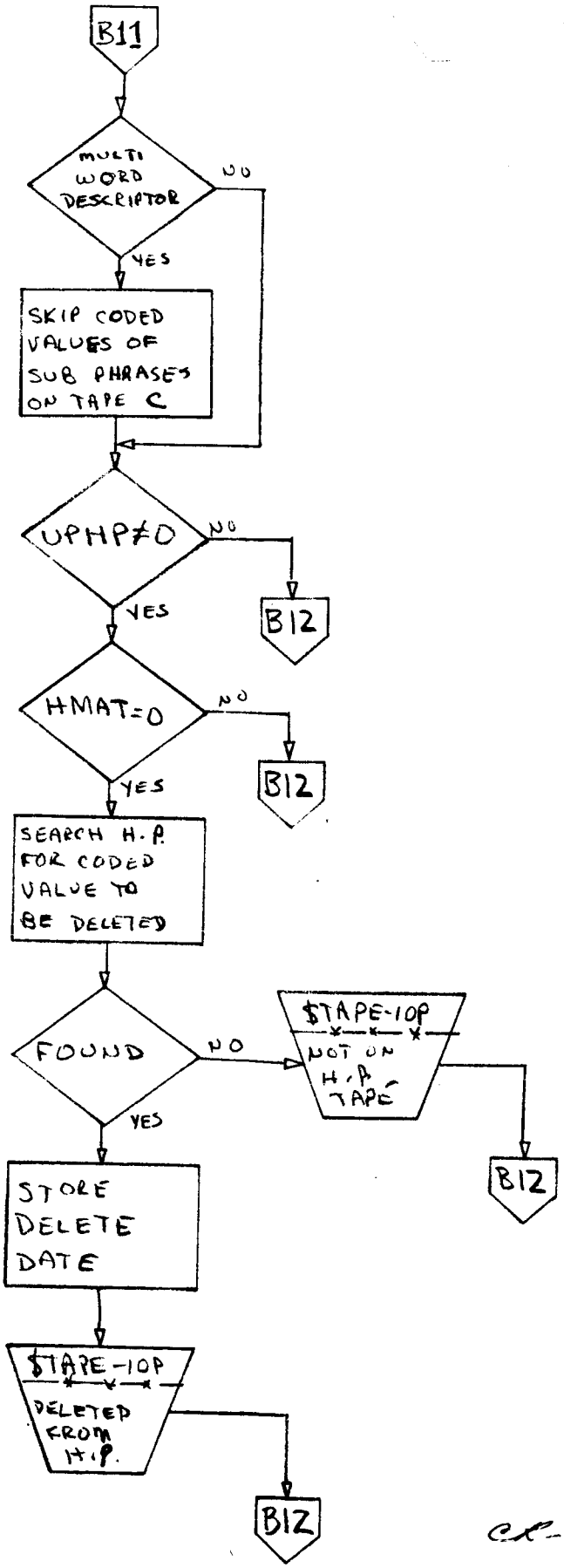
CR-62021


B9
$TAPE-IOP
DESCRIPTOR
. . .
CONVERT CODED VALUE TO BCD FORMAT
OLD UBER TAPE
NO
YES
$TAPE-IOP
DELETE NOT PROCESSED
B12
USER ON OLD USER TAPE
NO
YES
RNT 10
NO
1
YES
SPECIAL
DESCR. ON TAPE
YES
NO
REMOVE 5 OR 6 FROM CODED VALUE IF SINGLE OR MULTI-WORD DESCR
STORE INDICATORS
LOAD INDICATORS WITH MATCH WORD FROM LIST
$TAPE-IOP
NOT ON TAPE
B12
B11
ADD 1 TO L OF L,M,N WORD
RFT 2
B11
1
SINGLE WORD DESCR.
NO
YES
B10
INCLEMENT DELETED DESCRIPTOR COUNTER BY 1
REMOVE BIT 35 FROM WORD ON LIST
$TAPE-IOP
DELETED

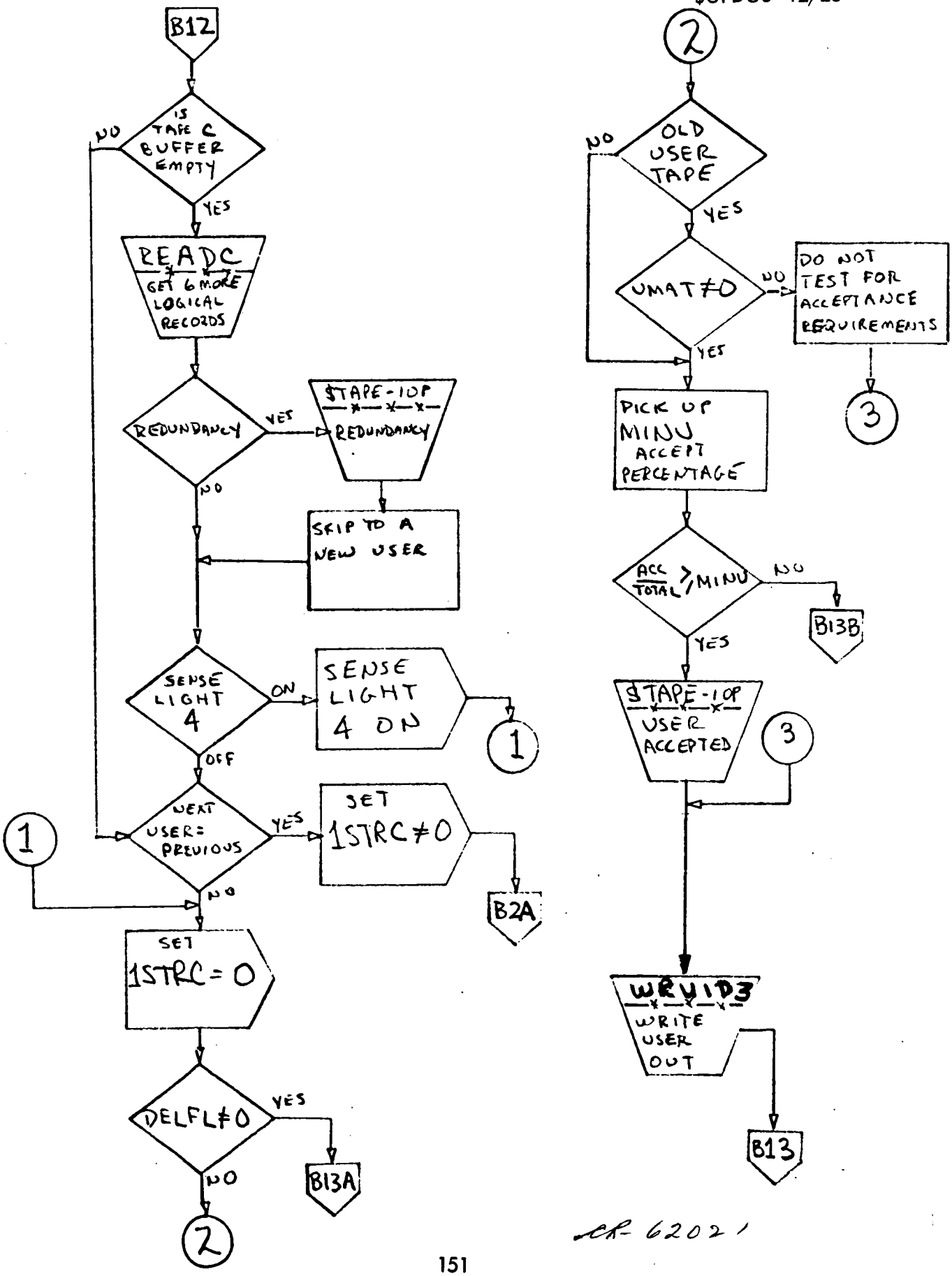
CR-620.21


B10
DESCR FOUND ON PHRASE LIST
NO
$TAPE-IOP
CANNOT DELETE
B11
YES
REMOVE CODED VALUE FROM PHRASE LIST LEAVING ONLY FIRST & LAST 5 BITS
$TAPE-IOP
DELETED
B11

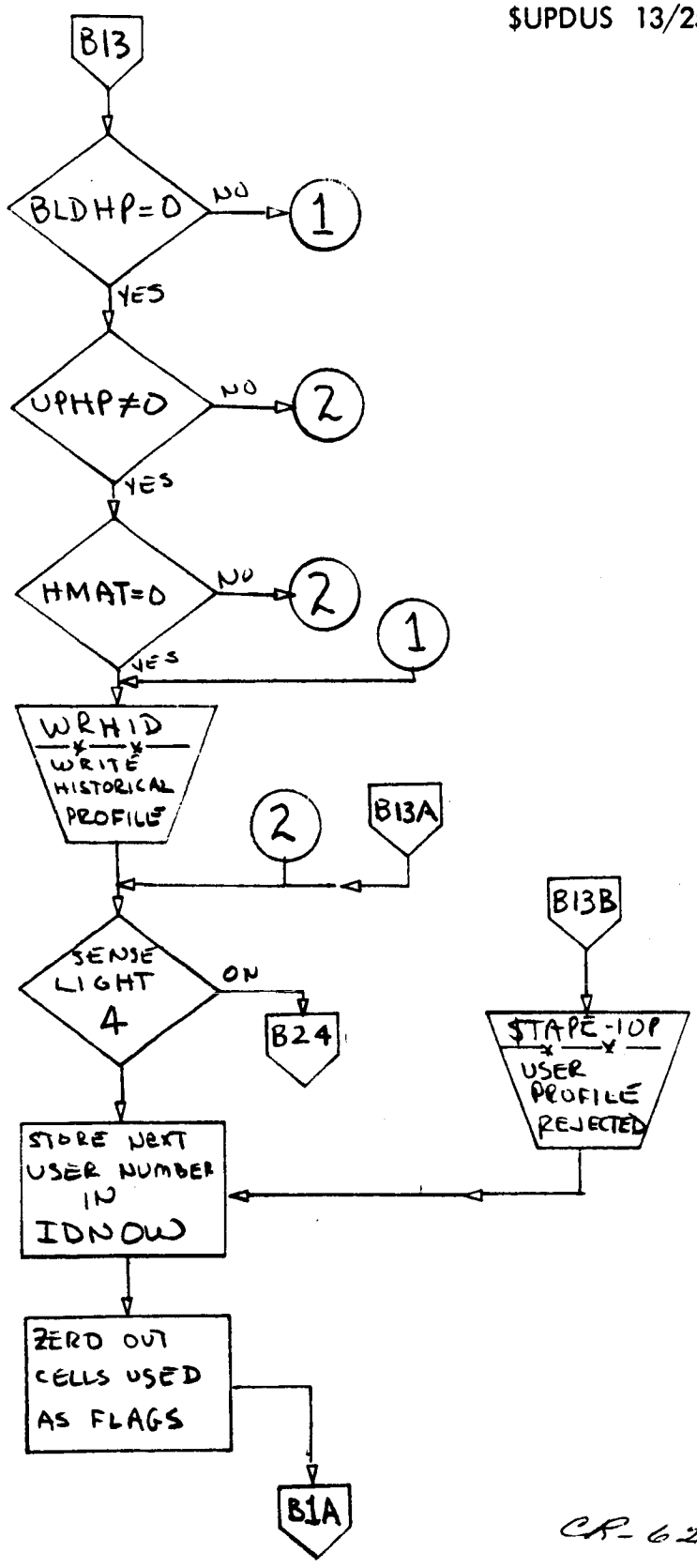
CR-62021


B11
MULTI WORD DESCRIPTOR
NO
YES
SKIP CODED VALUES OF SUB PHRASES ON TAPE C
UPHP≠0
NO
YES
B12
HMAT=0
NO
YES
B12
SEARCH H.P. FOR CODED VALUE TO BE DELETED
FOUND
NO
YES
$TAPE-IOP
NOT ON H.P. TAPE
B12
STORE
DELETE
DATE
$TAPE-IOP
DELETED FROM H.P.
B12

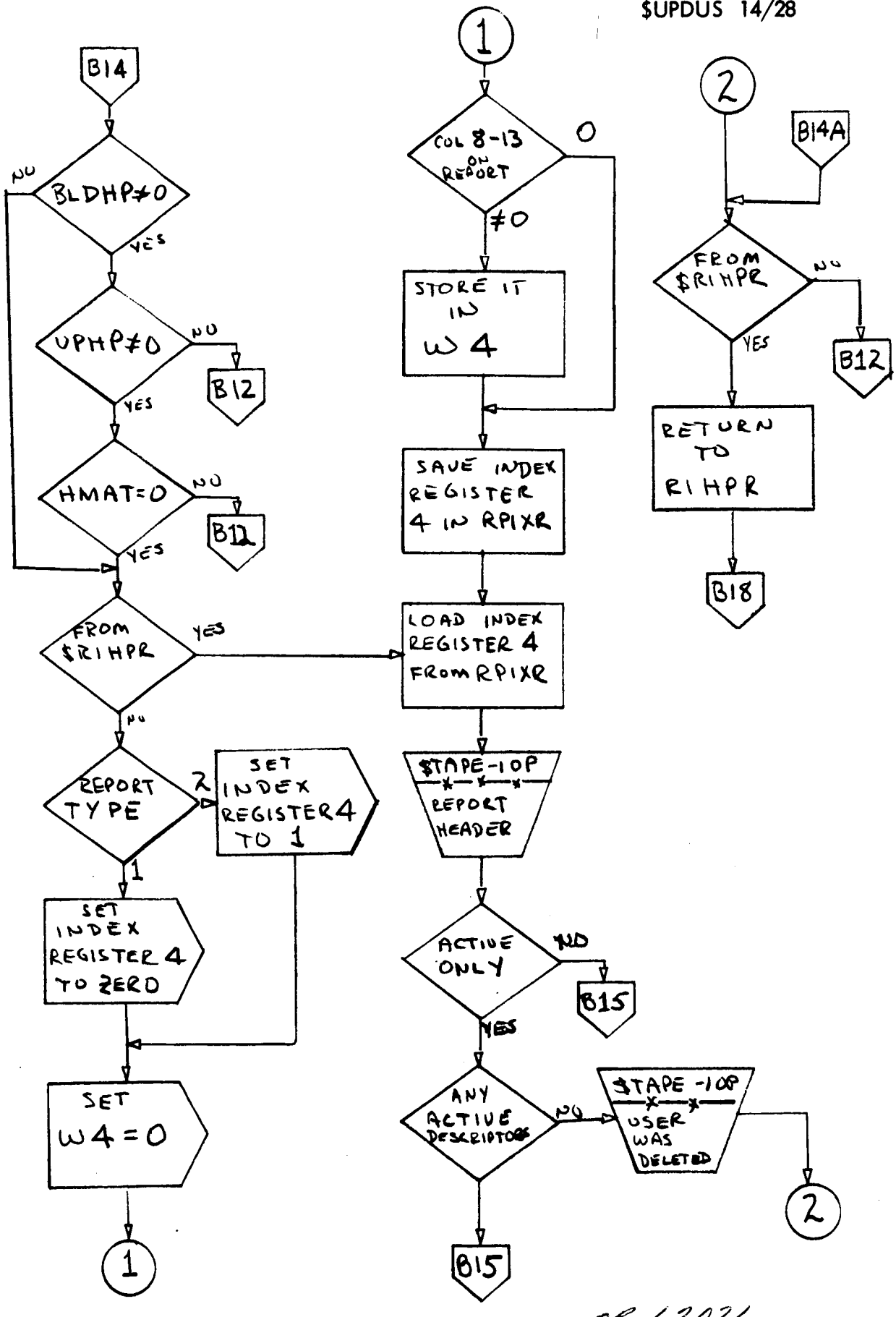
CR-62021


B12
IS TAPE C BUFFER EMPTY
NO
YES
READC
GET 6 MORE LOGICAL RECORDS
REDUNDANCY
YES
$TAPE-IOP
REDUNDANCY
NO
SKIP TO A NEW USER
SENSE LIGHT 4
ON
SENSE LIGHT 4 ON
1
OFF
NEXT USER= PREVIOUS
YES
SET 1STRC≠0
B2A
1
NO
SET 1STRC=0
DELFL≠0
YES
B13A
NO
2
2
OLD USER TAPE
NO
YES
UMAT≠0
NO
DO NOT TEST FOR ACCEPTANCE REQUIREMENTS
3
YES
PICK UP MINU ACCEPT PERCENTAGE
ACC TOTAL ≥ MINU
NO
B13B
YES
$TAPE-IOP
USER ACCEPTED
3
WRUID3
WRITE USER OUT
B13

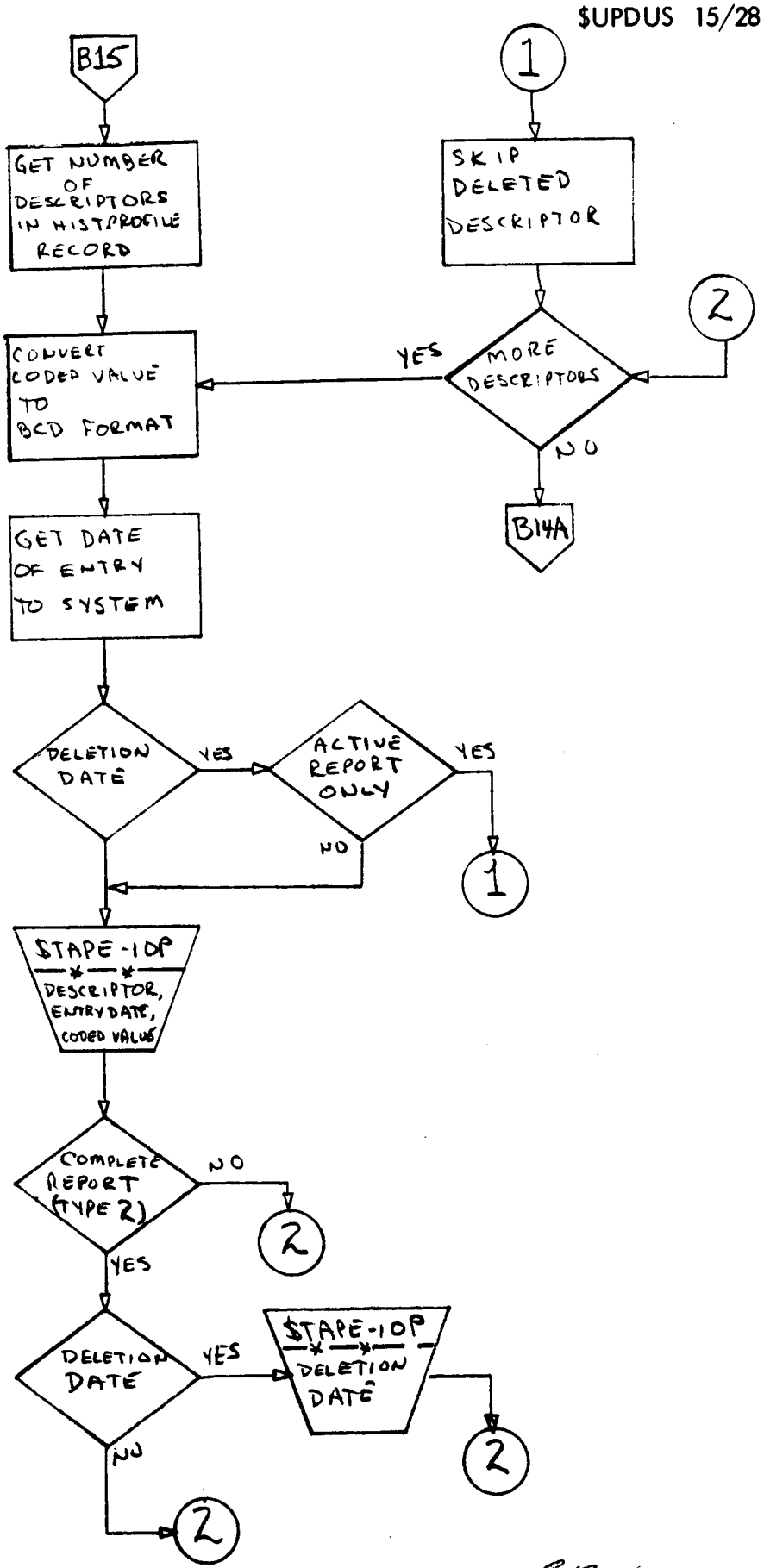
CR 62021


B13
BLDHP=0
NO
1
YES
UPHP≠0
NO
2
YES
HMAT=0
NO
2
1
YES
WRHID
WRITE HISTORICAL PROFILE
2
B13A
B13B
SENSE LIGHT 4
ON
B24
STAPE-10P
USER PROFILE REJECTED
STORE NEXT USER NUMBER IN IDNOW
ZERO OUT CELLS USED AS FLAGS
B1A

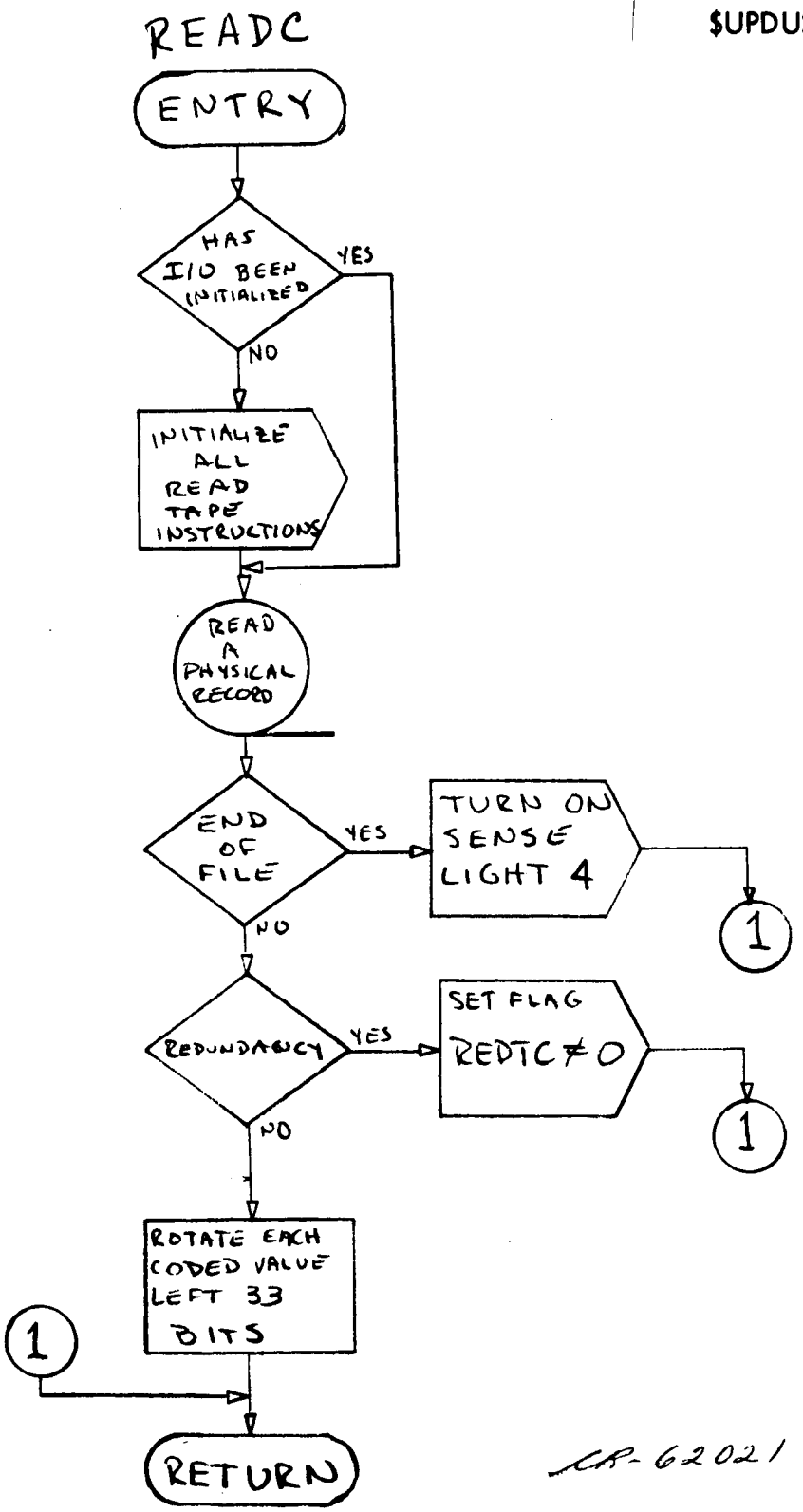
CR-62021


B14
BLDHP≠0
NO
YES
UPHP≠0
NO
B12
YES
HMAT=0
NO
B12
YES
FROM $RIHPR
YES
NO
REPORT TYPE
2
SET INDEX REGISTER 4 TO 1
1
SET INDEX REGISTER 4 TO ZERO
SET W4=0
1
1
COL 8-13 ON REPORT
0
≠0
STORE IT IN W4
SAVE INDEX REGISTER 4 IN RPIXR
LOAD INDEX REGISTER 4 FROM RPIXR
$TAPE-IOP
REPORT HEADER
ACTIVE ONLY
NO
B15
YES
ANY ACTIVE DESCRIPTORS
NO
$TAPE-IOP
USER WAS DELETED
2
B15
2
B14A
FROM $RIHPR
NO
B12
YES
RETURN TO RIHPR
B18

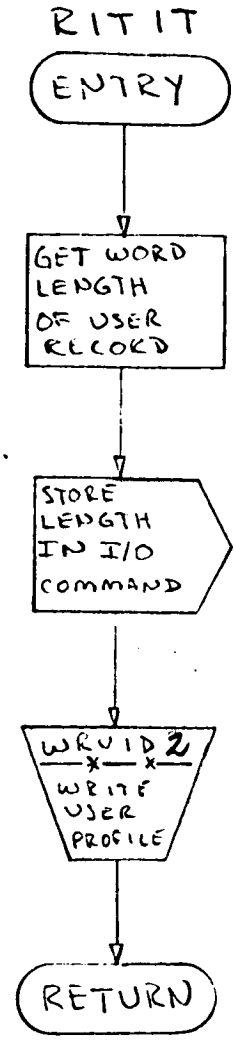
CR-62021


B15
GET NUMBER OF DESCRIPTORS IN HISTPROFILE RECORD
CONVERT CODED VALUE TO BCD FORMAT
GET DATE OF ENTRY TO SYSTEM
DELETION DATE
YES
ACTIVE REPORT ONLY
YES
NO
1
$TAPE-IOP
DESCRIPTOR, ENTRY DATE, CODED VALUE
COMPLETE REPORT (TYPE 2)
NO
2
YES
DELETION DATE
YES
$TAPE-IOP
DELETION DATE
2
NO
2
1
SKIP DELETED DESCRIPTOR
2
MORE DESCRIPTORS
YES
NO
B14A

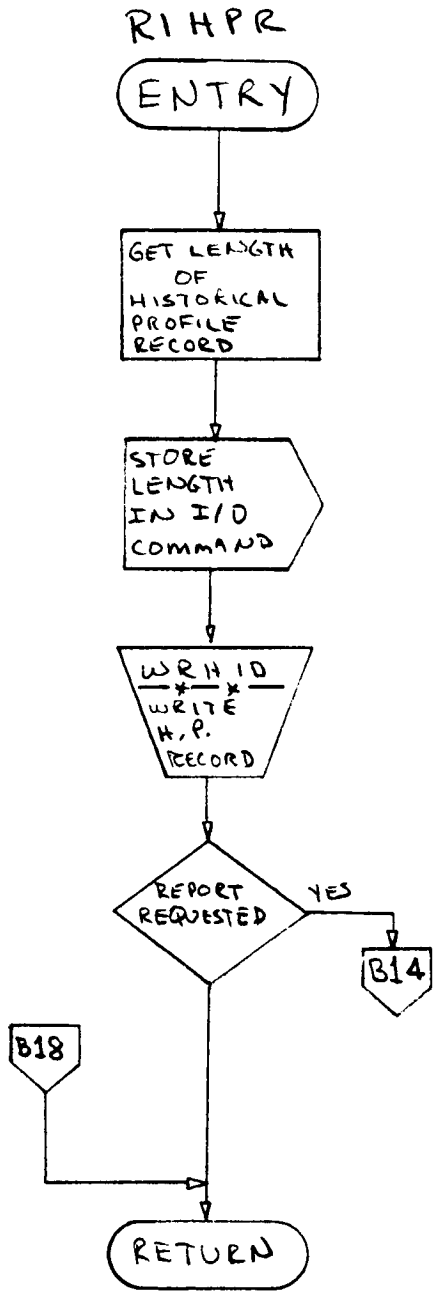
CR 6202)


READC
ENTRY
HAS I/O BEEN INITIALIZED
YES
NO
INITIALIZE ALL READ TAPE INSTRUCTIONS
READ A PHYSICAL RECORD
END OF FILE
YES
TURN ON SENSE LIGHT 4
1
NO
REDUNDANCY
YES
SET FLAG REDTC≠0
1
NO
ROTATE EACH CODED VALUE LEFT 33 BITS
1
RETURN

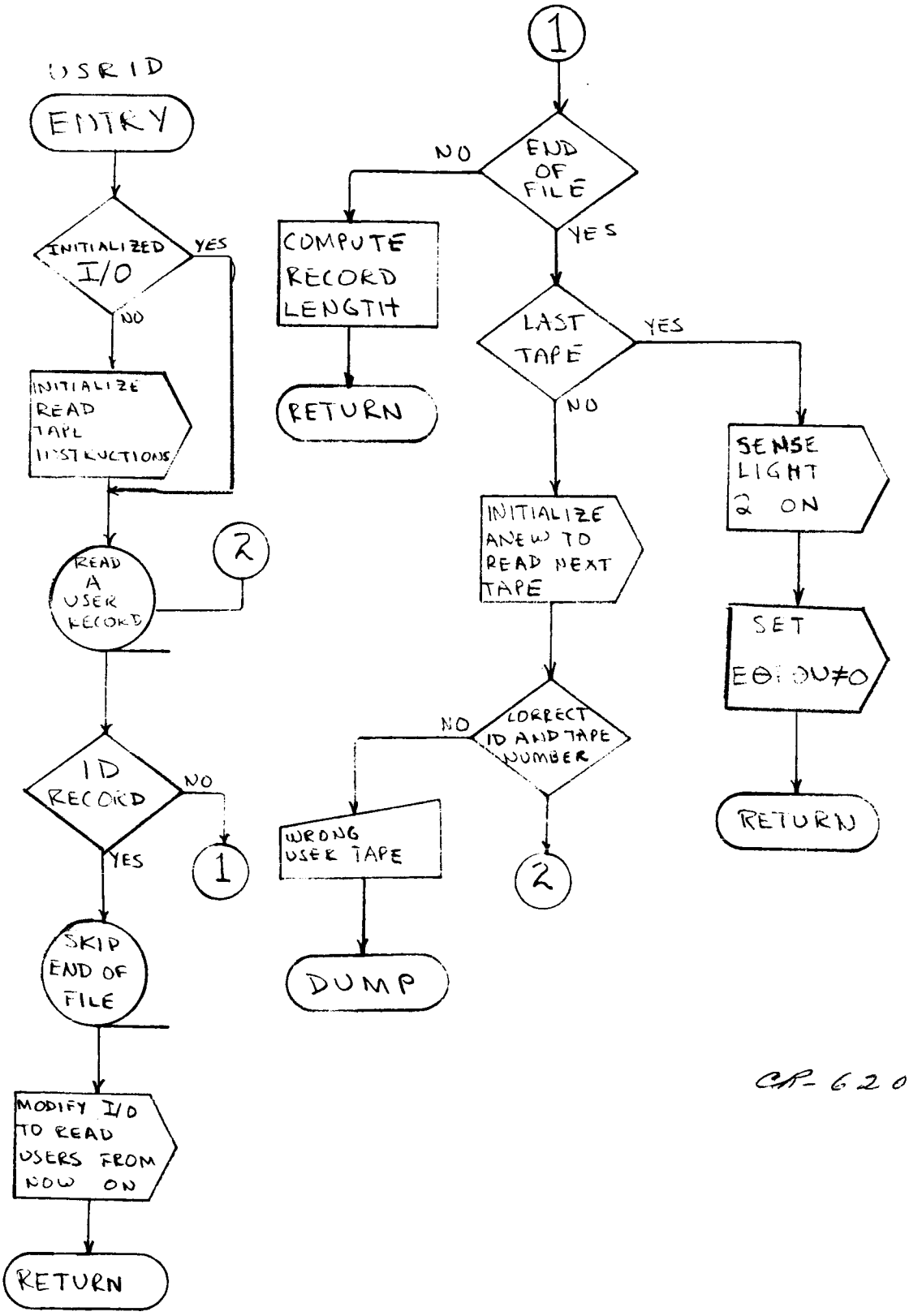
CR-62021


RIT IT
ENTRY
GET WORD LENGTH OF USER RECORD
STORE LENGTH IN I/O COMMAND
WRVID 2
WRITE USER PROFILE
RETURN

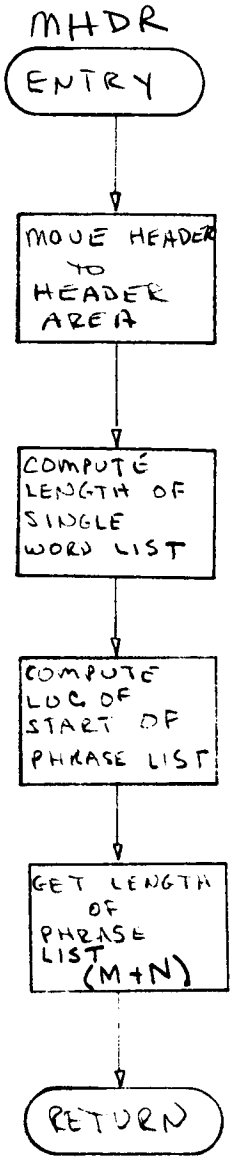
CR-62021


R1HPR
ENTRY
GET LENGTH OF HISTORICAL PROFILE RECORD
STORE LENGTH IN I/O COMMAND
WRITE H.P. RECORD
REPORT REQUESTED
YES
B14
B18
RETURN

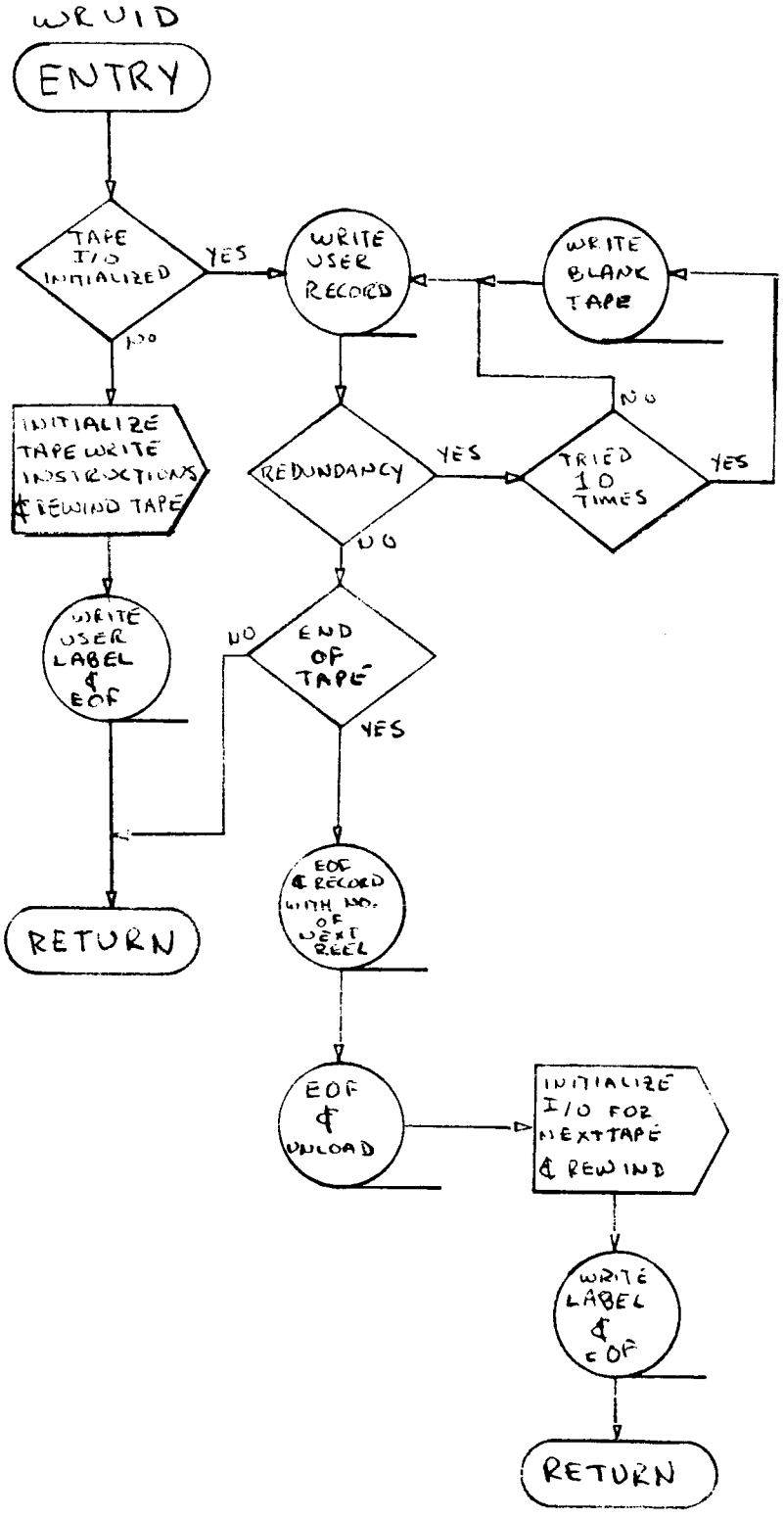
CR-62021


USRID
ENTRY
INITIALIZED I/O
YES
NO
INITIALIZE READ TAPE INSTRUCTIONS
READ A USER RECORD
2
ID RECORD
NO
YES
1
SKIP END OF FILE
MODIFY I/O TO READ USERS FROM NOW ON
RETURN
1
END OF FILE
NO
YES
COMPUTE RECORD LENGTH
RETURN
LAST TAPE
YES
NO
INITIALIZE ANEW TO READ NEXT TAPE
CORRECT ID AND TAPE NUMBER
NO
WRONG USER TAPE
DUMP
2
SENSE LIGHT 2 ON
SET EOF DU≠0
RETURN

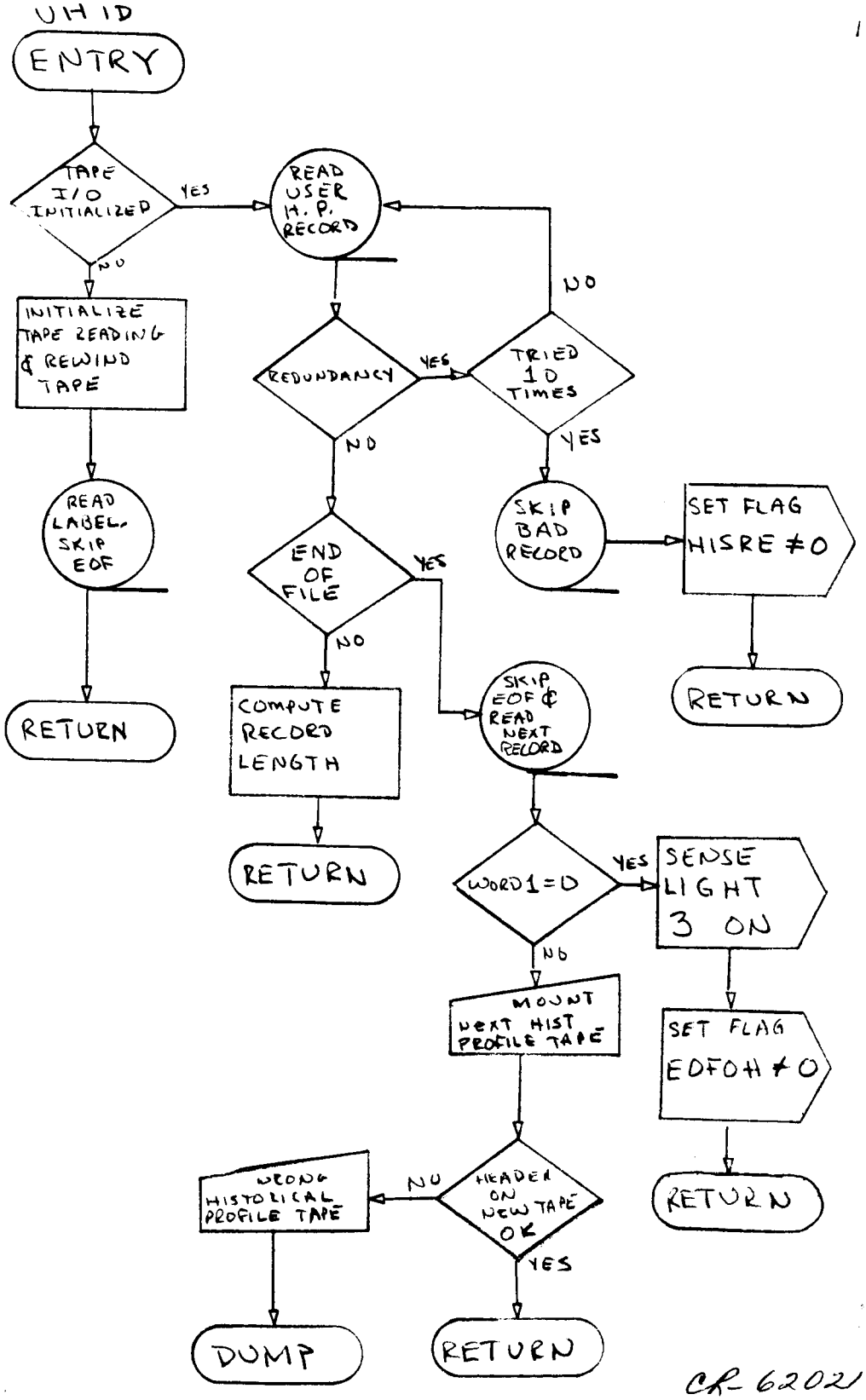
CR-62021


MHDR
ENTRY
MOVE HEADER TO HEADER AREA
COMPUTE LENGTH OF SINGLE WORD LIST
COMPUTE LOC OF START OF PHRASE LIST
GET LENGTH OF PHRASE LIST (M+N)
RETURN

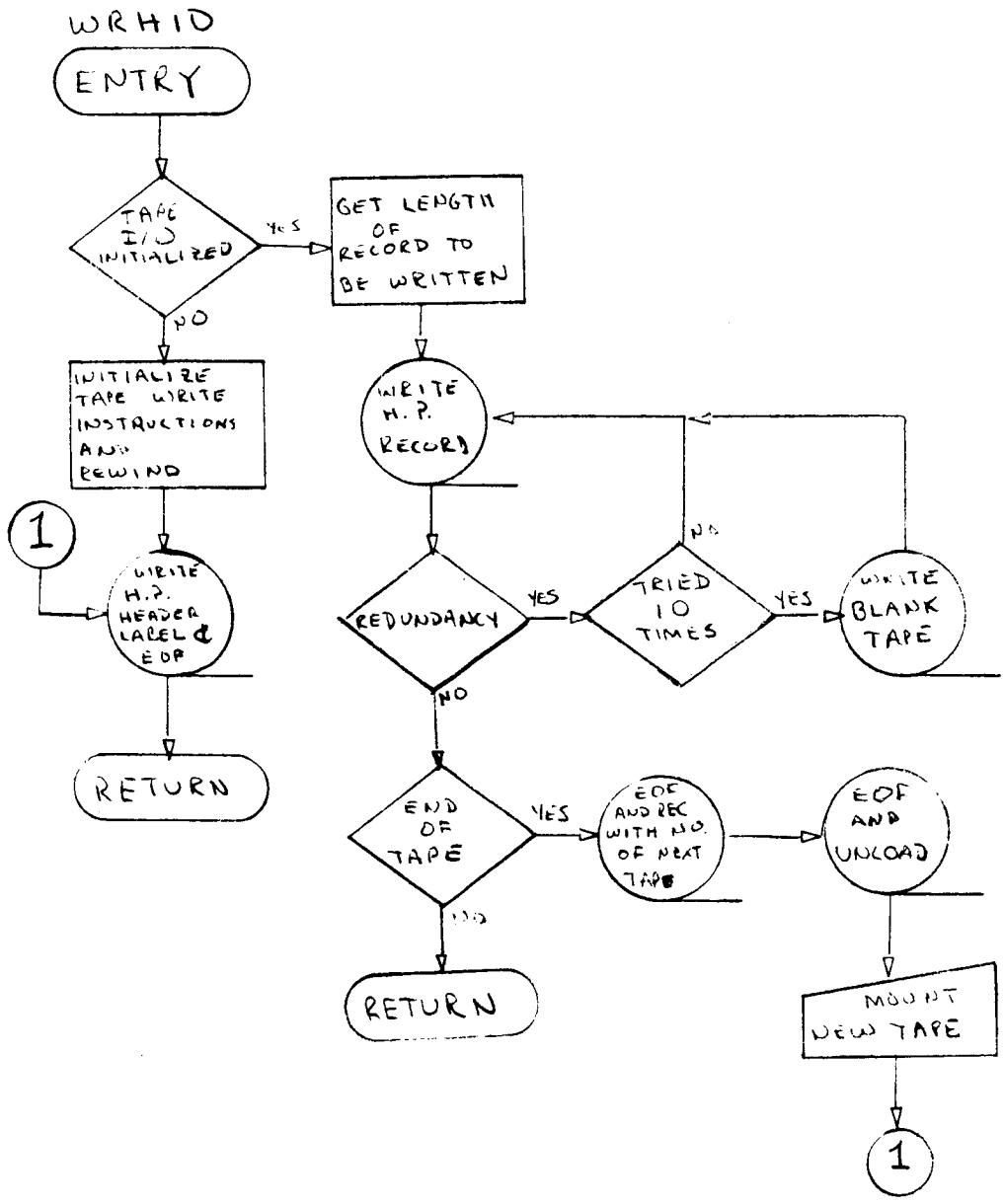
CR-6202/


WRUID
ENTRY
TAPE I/O INITIALIZED
YES
NO
WRITE USER RECORD
WRITE BLANK TAPE
INITIALIZE TAPE WRITE INSTRUCTIONS & REWIND TAPE
REDUNDANCY
YES
NO
TRIED 10 TIMES
YES
NO
WRITE USER LABEL & EOF
NO
END OF TAPE
YES
RETURN
EOF & RECORD WITH NO. OF NEXT REEL
EOF & UNLOAD
INITIALIZE I/O FOR NEXT TAPE & REWIND
WRITE LABEL & EOF
RETURN

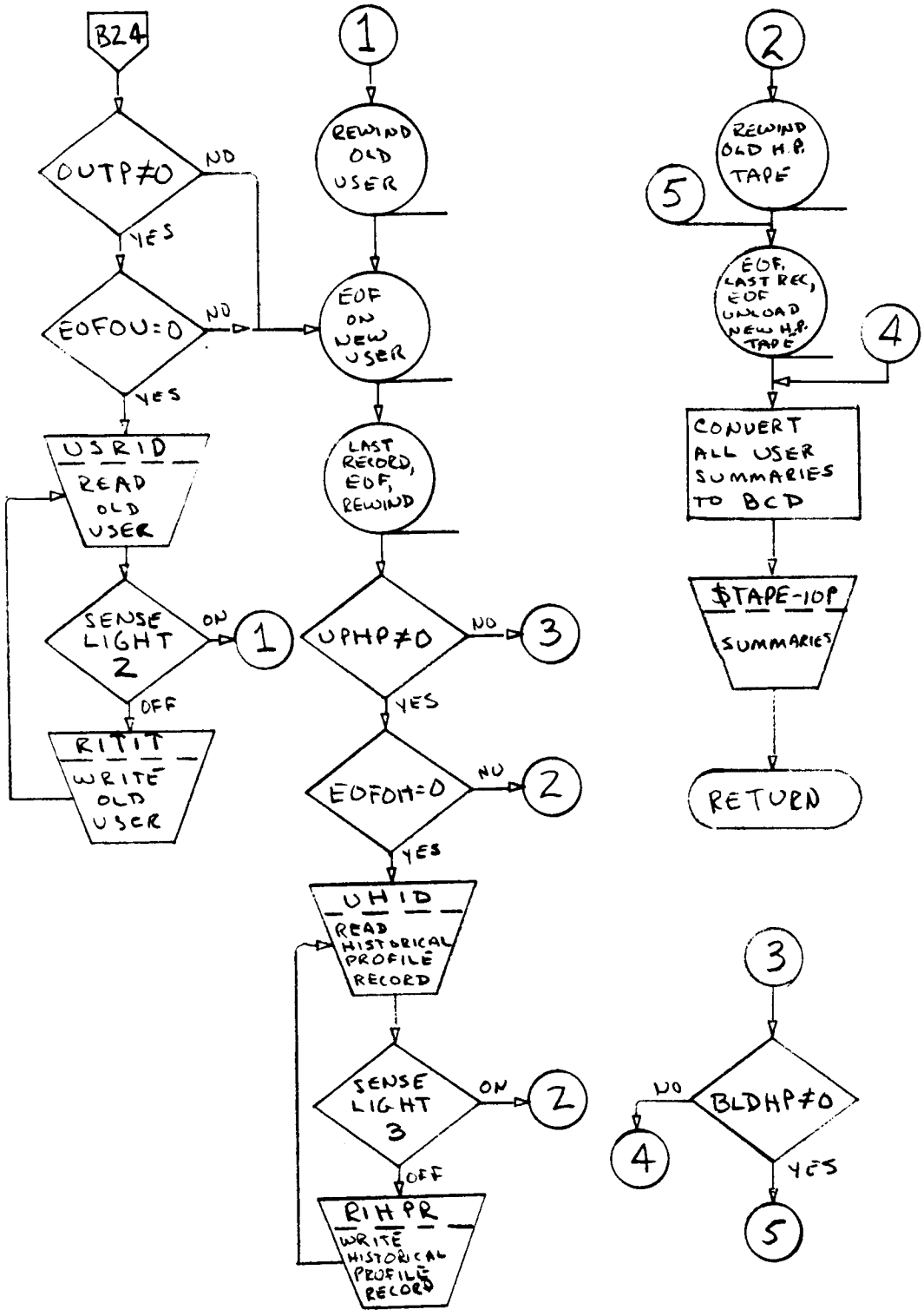
CR-62021


UH ID
ENTRY
TAPE I/O INITIALIZED
YES
NO
READ USER M. P. RECORD
INITIALIZE TAPE READING & REWIND TAPE
READ LABEL, SKIP EOF
RETURN
REDUNDANCY
YES
NO
TRIED 10 TIMES
NO
YES
END OF FILE
YES
NO
SKIP BAD RECORD
SET FLAG HISRE ≠ 0
RETURN
COMPUTE RECORD LENGTH
RETURN
SKIP EOF & READ NEXT RECORD
WORD1 = 0
YES
NO
SENSE LIGHT 3 ON
SET FLAG EOFOH ≠ 0
RETURN
MOUNT NEXT HIST PROFILE TAPE
HEADER ON NEW TAPE OK
NO
YES
WRONG HISTORICAL PROFILE TAPE
DUMP
RETURN

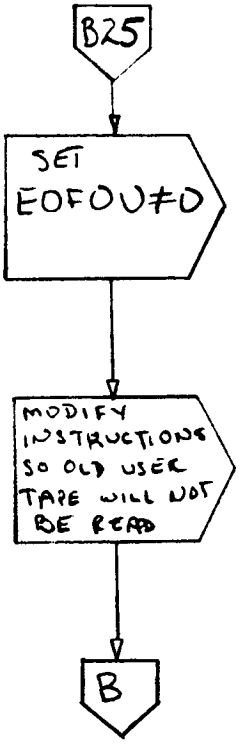
CR-6202

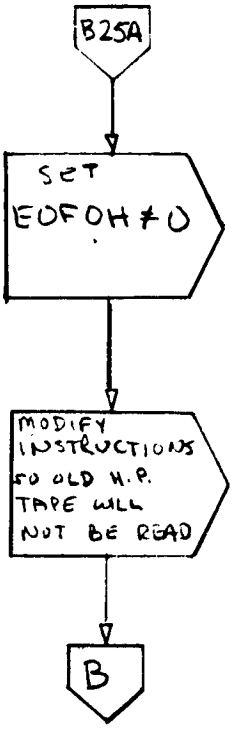

WRHID
ENTRY
TAPE I/O INITIALIZED
YES
NO
GET LENGTH OF RECORD TO BE WRITTEN
INITIALIZE TAPE WRITE INSTRUCTIONS AND REWIND
WRITE H.P. RECORD
1
WRITE H.P. HEADER LABEL & EOF
REDUNDANCY
YES
NO
TRIED 10 TIMES
YES
NO
WRITE BLANK TAPE
RETURN
END OF TAPE
YES
NO
EOF AND REC WITH NO. OF NEXT TAPE
EOF AND UNLOAD
MOUNT NEW TAPE
RETURN
1

CR-62021

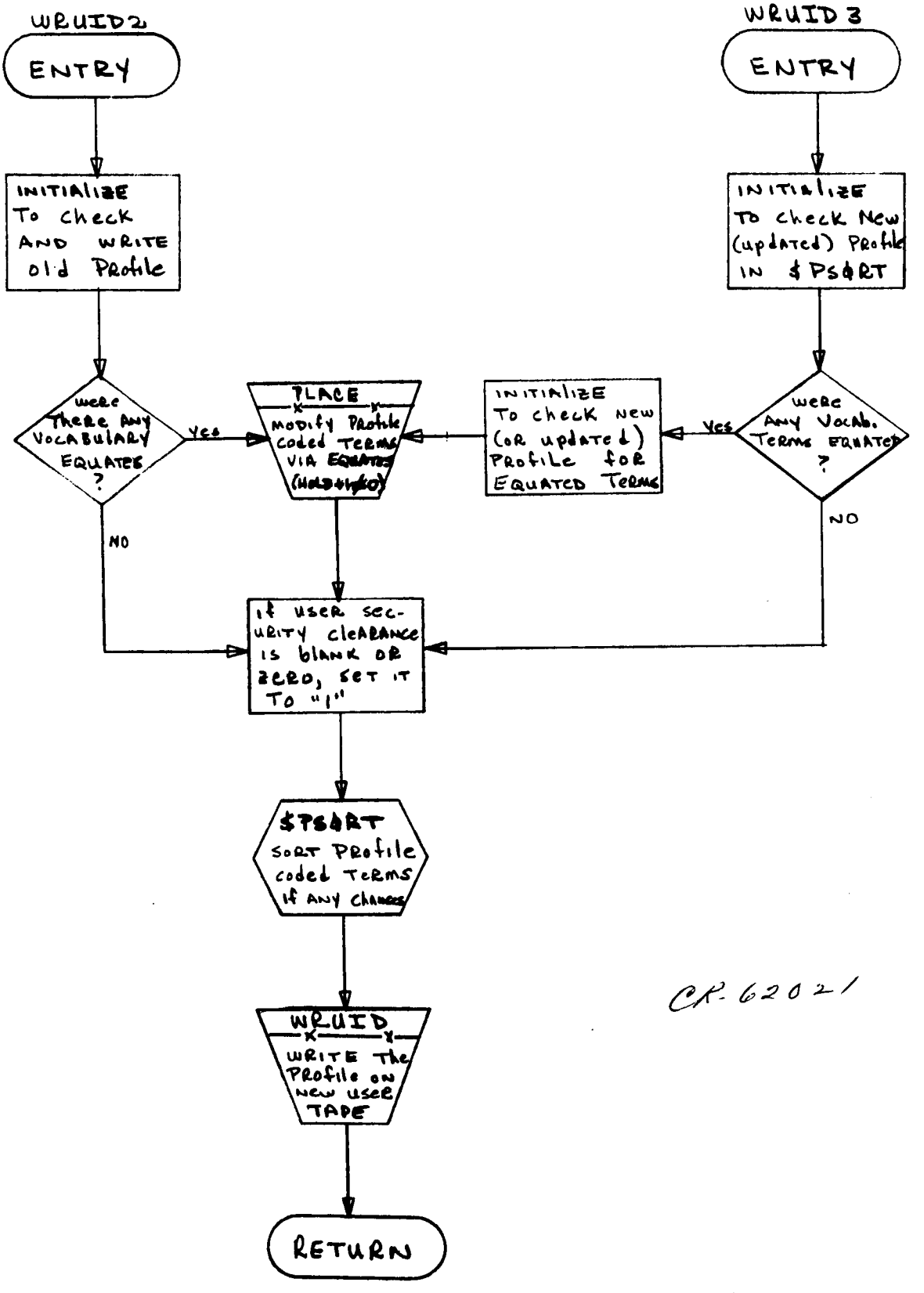

B24
OUTP≠0
NO
YES
EOFOU=0
NO
YES
USRID
READ OLD USER
SENSE LIGHT 2
ON
1
OFF
RITIT
WRITE OLD USER
1
REWIND OLD USER
EOF ON NEW USER
LAST RECORD, EOF, REWIND
UPHP≠0
NO
3
YES
EOFOM=0
NO
2
YES
UHID
READ HISTORICAL PROFILE RECORD
SENSE LIGHT 3
ON
2
OFF
RIHPR
WRITE HISTORICAL PROFILE RECORD
2
REWIND OLD H.P. TAPE
5
EOF, LAST REC, EOF UNLOAD NEW H.P. TAPE
4
CONVERT ALL USER SUMMARIES TO BCD
$TAPE-IOP
SUMMARIES
RETURN
3
NO
BLDHP≠0
4
YES
5

CR-62021

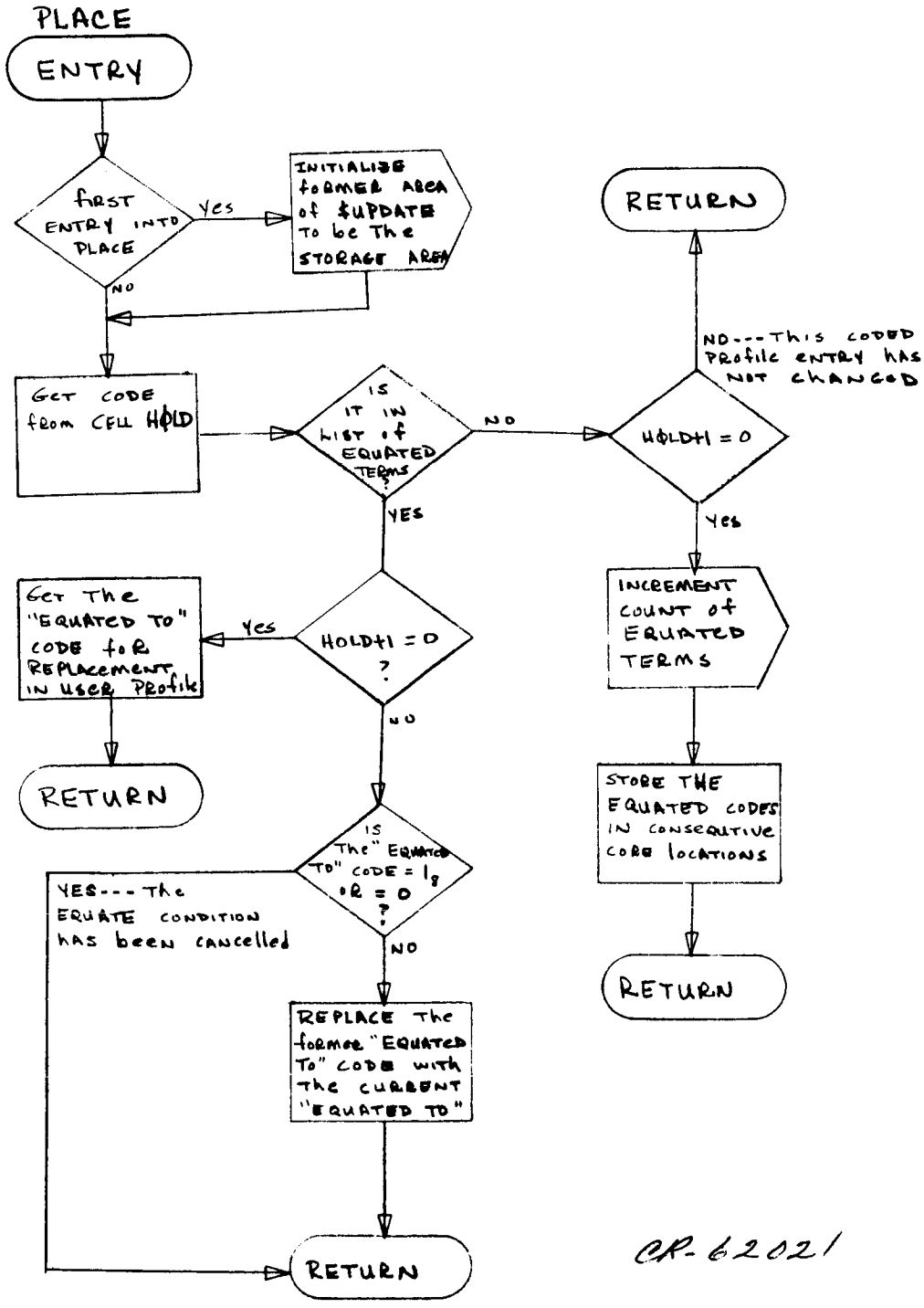

B25
SET
EOFOU≠0
MODIFY
INSTRUCTIONS
SO OLD USER
TAPE WILL NOT
BE READ
B


B25A
SET
EOFOH≠0
MODIFY
INSTRUCTIONS
SO OLD H.P.
TAPE WILL
NOT BE READ
B

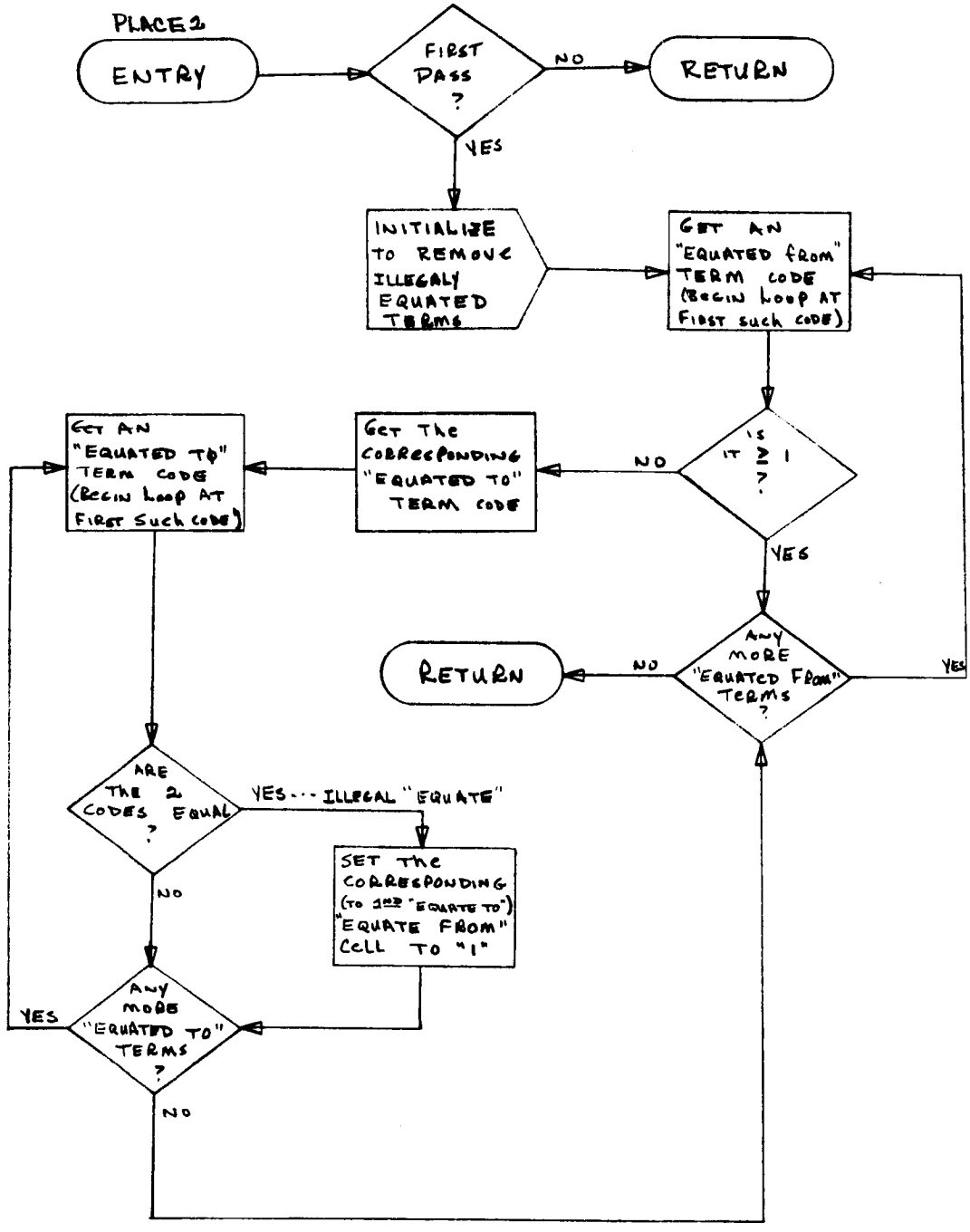
CR-02021


WRUID2
ENTRY
INITIALIZE To check AND WRITE old Profile
were There ANY VOCABULARY EQUATES ?
Yes
NO
PLACE
X
Y
MODify Profile coded TERMS VIA EQUATES
INITIALIZE To check NEW (OR updated) Profile for EQUATED TERMS
WRUID 3
ENTRY
INITIALIZE To check New (updated) Profile IN PSRT
were ANY VOCAB. TERMS EQUATED ?
Yes
NO
If USER SECURITY CLEARANCE IS BLANK OR ZERO, SET IT TO "1"
PSRT
SORT Profile coded TERMS if ANY CHANGES
WRUID
X
Y
WRITE The PROFILE ON NEW USER TAPE
RETURN

CR-62021


PLACE
ENTRY
FIRST ENTRY INTO PLACE
Yes
INITIALISE former AREA of $UPDATE To be The STORAGE AREA
NO
Get CODE from CELL HOLD
IS IT IN LIST of EQUATED TERMS ?
NO
HOLD+1 = 0
Yes
RETURN
NO---This CODED Profile ENTRY has NOT CHANGED
YES
Get The "EQUATED TO" CODE for REPLACEMENT IN USER Profile
Yes
HOLD+1 = 0 ?
INCREMENT COUNT of EQUATED TERMS
RETURN
NO
STORE THE EQUATED CODES IN CONSECUTIVE CORE LOCATIONS
IS The "EQUATED TO" CODE = 1₈ OR = 0 ?
YES--- The EQUATE CONDITION has been cancelled
NO
RETURN
REPLACE The former "EQUATED TO" CODE with The CURRENT "EQUATED TO"
RETURN

CR-62021

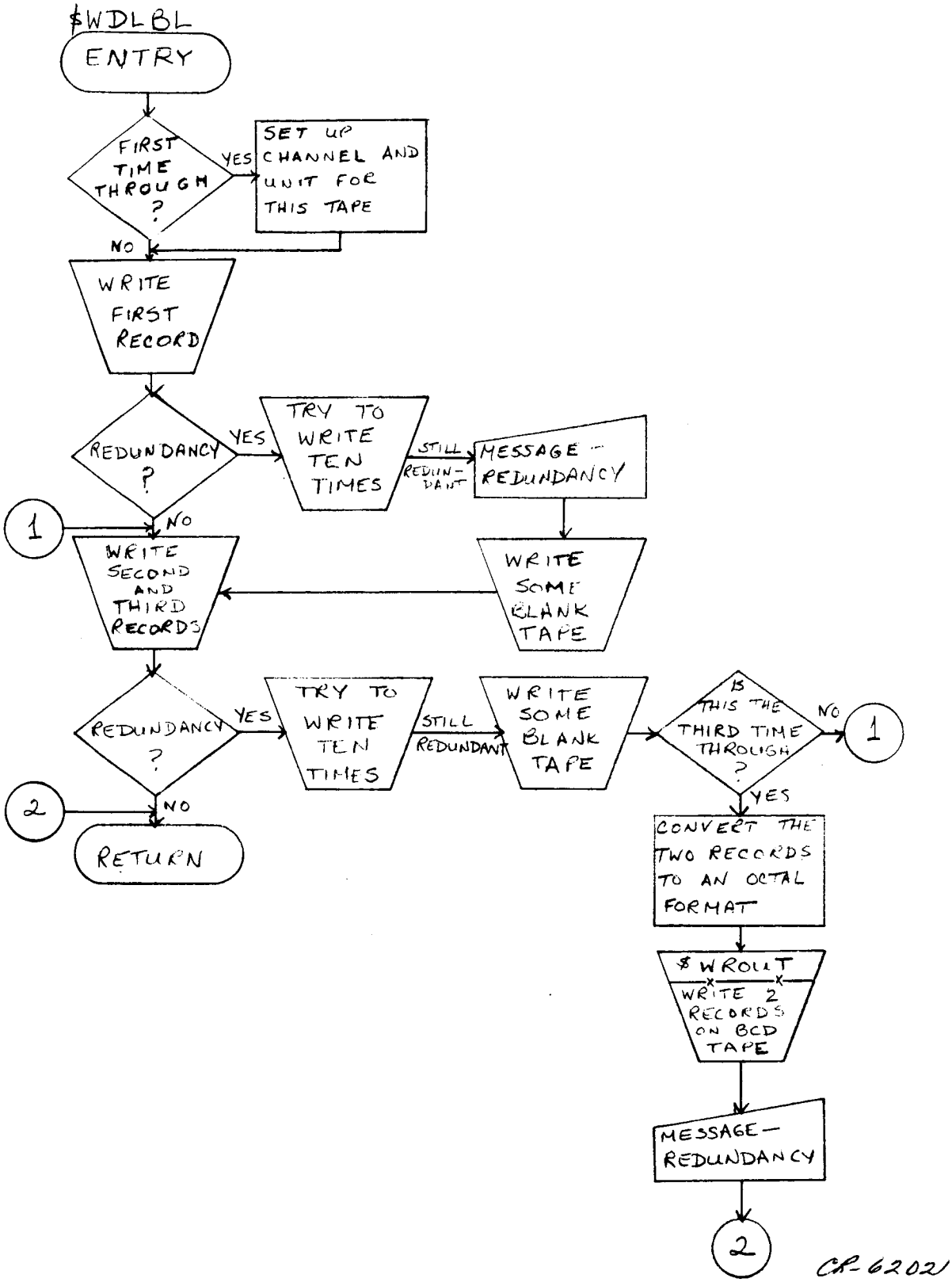

PLACE 2
ENTRY
FIRST PASS ?
NO
RETURN
YES
INITIALIZE TO REMOVE ILLEGALY EQUATED TERMS
GET AN "EQUATED FROM" TERM CODE (BEGIN LOOP AT FIRST SUCH CODE)
IS IT ≥ 1 ?
NO
GET THE CORRESPONDING "EQUATED TO" TERM CODE
GET AN "EQUATED TO" TERM CODE (BEGIN LOOP AT FIRST SUCH CODE)
YES
ANY MORE "EQUATED FROM" TERMS ?
NO
RETURN
YES
ARE THE 2 CODES EQUAL ?
YES --- ILLEGAL "EQUATE"
SET THE CORRESPONDING (TO 1ST "EQUATE TO") "EQUATE FROM" CELL TO "1"
NO
ANY MORE "EQUATED TO" TERMS ?
YES
NO

CR-62021

Subroutine WDLBL

This subroutine writes the first file of the new vocabulary. The first record contains the tape label. If there is a redundancy, an attempt is made to write the record ten times. If unsuccessful, a message indicating the redundancy condition is printed and tape is spaced forward. The second record, which contains the catalog of minimum and maximum entries, and the third record, which contains a table of the number of logical records per file, are written next. If there is a redundancy, an attempt is made to write the records ten times. If the redundancy persists, some blank tape is written, and an attempt is made to write the two records again. If the redundancy still exists, one more try is made writing blank tape and writing the records. If there is still a redundancy, the records are written on the system output tape with twelve octal numbers for each computer word.

CR-62021

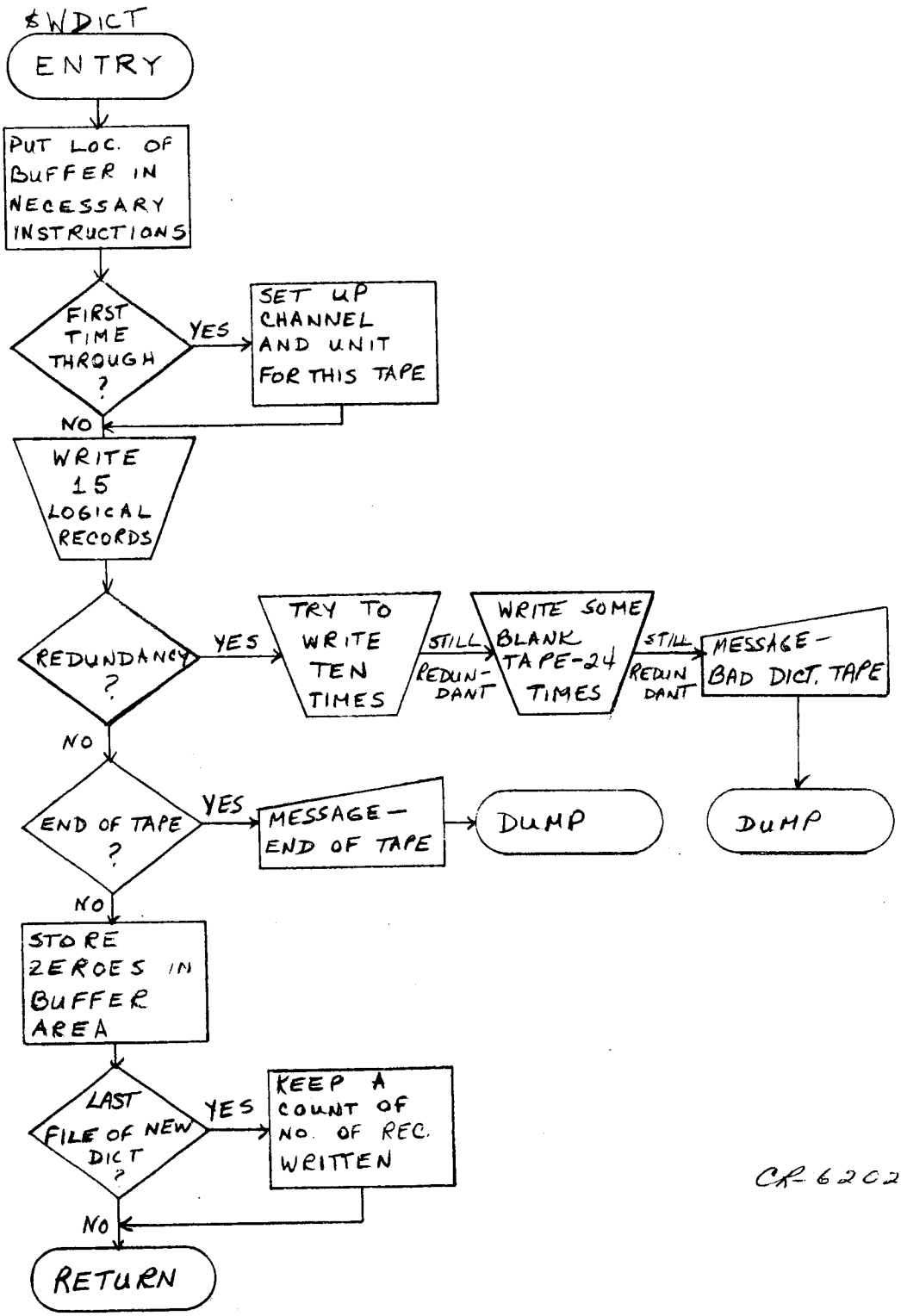

$WDLBL
ENTRY
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE FIRST RECORD
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
MESSAGE - REDUNDANCY
WRITE SOME BLANK TAPE
1
NO
WRITE SECOND AND THIRD RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
WRITE SOME BLANK TAPE
IS THIS THE THIRD TIME THROUGH ?
NO
1
YES
CONVERT THE TWO RECORDS TO AN OCTAL FORMAT
$WROUT
WRITE 2 RECORDS ON BCD TAPE
MESSAGE - REDUNDANCY
2
2
NO
RETURN

CR-6202

Subroutine WDICT

This subroutine writes the new vocabulary descriptor records, which are blocked fifteen logical records per physical record. The procedures for a redundancy condition and an end-of-tape condition are the same as those used in WDOCT. A count of the number of records in the last file of the new vocabulary is kept for use by the subroutine UPDATE.

CR- 62021


$WDICT
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 15 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
WRITE SOME BLANK TAPE-24 TIMES
STILL REDUN-DANT
MESSAGE - BAD DICT. TAPE
DUMP
NO
END OF TAPE ?
YES
MESSAGE - END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
LAST FILE OF NEW DICT ?
YES
KEEP A COUNT OF NO. OF REC. WRITTEN
NO
RETURN

CR 62021

Subroutine WDOC

WDOC writes the binary document profiles tape. It is entered from subroutine PDTB. If a redundancy occurs while writing, ten attempts will be made, after which blank tape will be written. Ten more attempts will be made.

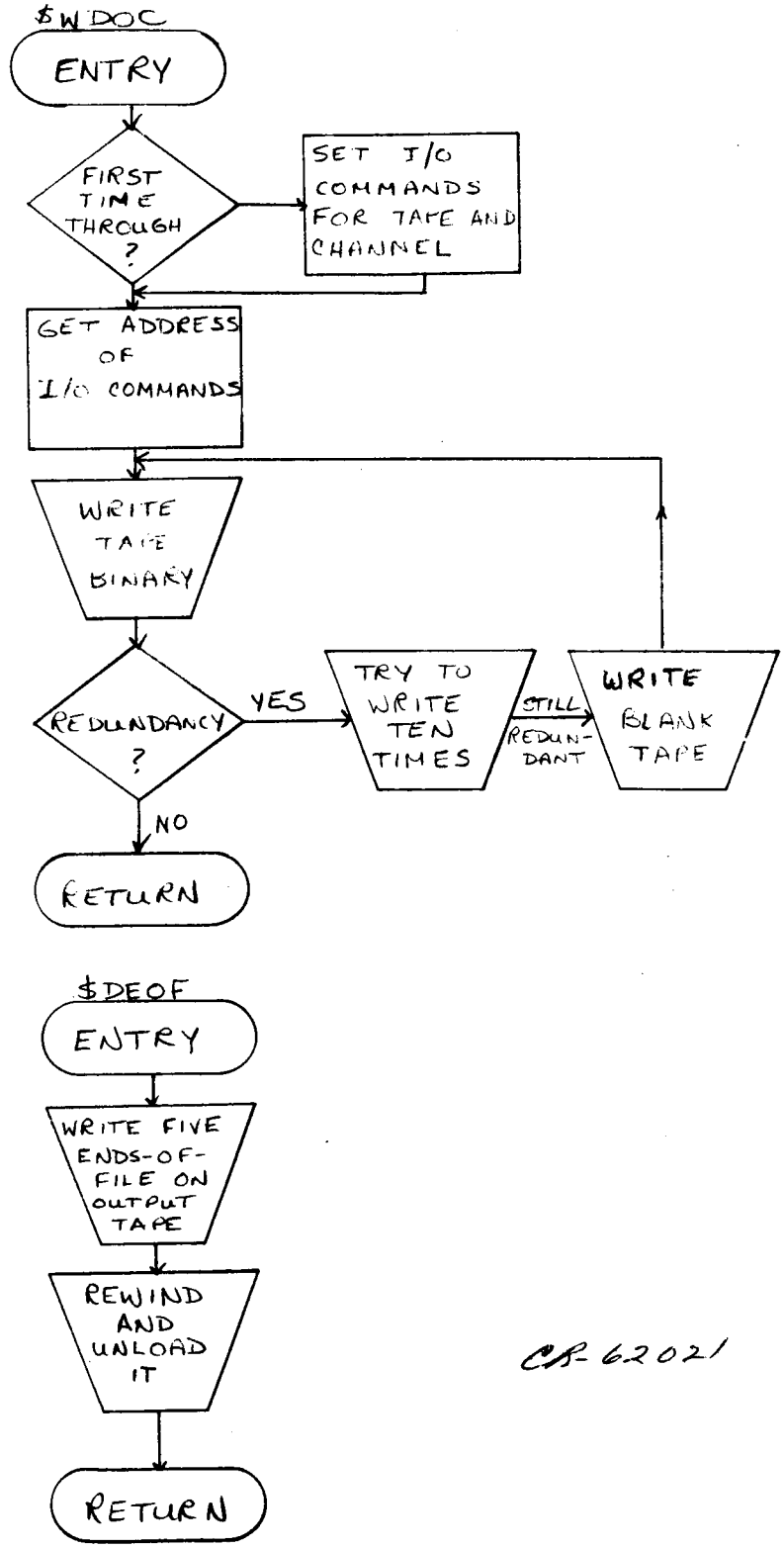
Subroutine DEOF

DEOF writes five EOF's on the binary document profiles tape. It then rewinds and unloads the tape.

CR-62021

$WDOC 1/1

$DEOF 1/1

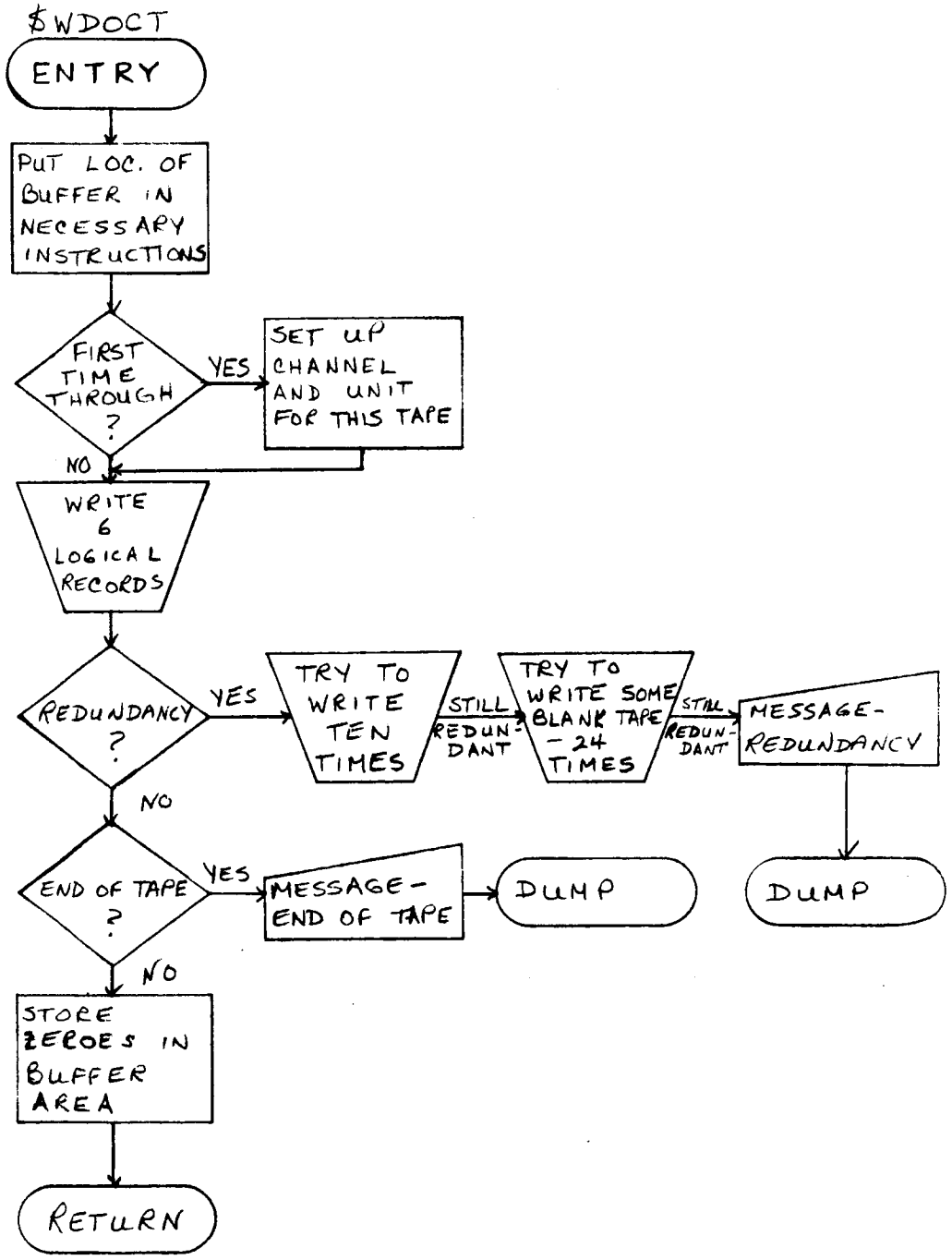


CR-62021

Subroutine WDOCT

This subroutine writes the coded document tape, which is blocked six logical records per physical record. For a redundant record, an attempt to write is made ten times. If the redundancy persists, blank tape is written and another attempt to write is made. This process can occur 24 times. If, finally, there is still a redundancy, a message is printed and a dump of core is taken. A message and a dump also will be given for an EOT.

CR-62021

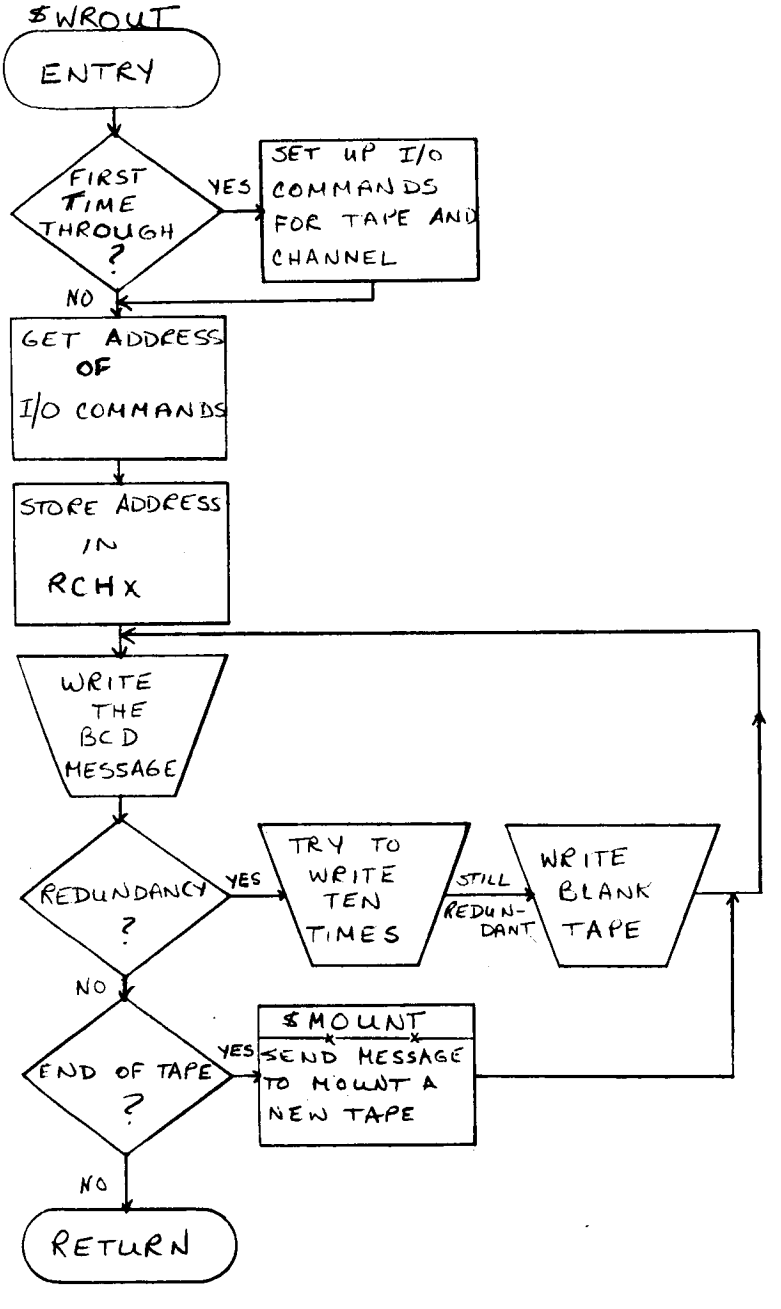

$WDOCT
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUNDANT
TRY TO WRITE SOME BLANK TAPE - 24 TIMES
STILL REDUNDANT
MESSAGE-REDUNDANCY
DUMP
NO
END OF TAPE ?
YES
MESSAGE-END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
RETURN

CR-62021

Subroutine WROUT

All messages written on the system output tape in PDTB are handled by this subroutine. The output instructions are initialized the first time through. The exception is the RCH instruction, which must be set up before each TSX to WROUT.

CR-62021


$WROUT
ENTRY
FIRST TIME THROUGH ?
YES
SET UP I/O COMMANDS FOR TAPE AND CHANNEL
NO
GET ADDRESS OF I/O COMMANDS
STORE ADDRESS IN RCHx
WRITE THE BCD MESSAGE
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
WRITE BLANK TAPE
NO
END OF TAPE ?
YES
$MOUNT
SEND MESSAGE TO MOUNT A NEW TAPE
NO
RETURN

CR-62021

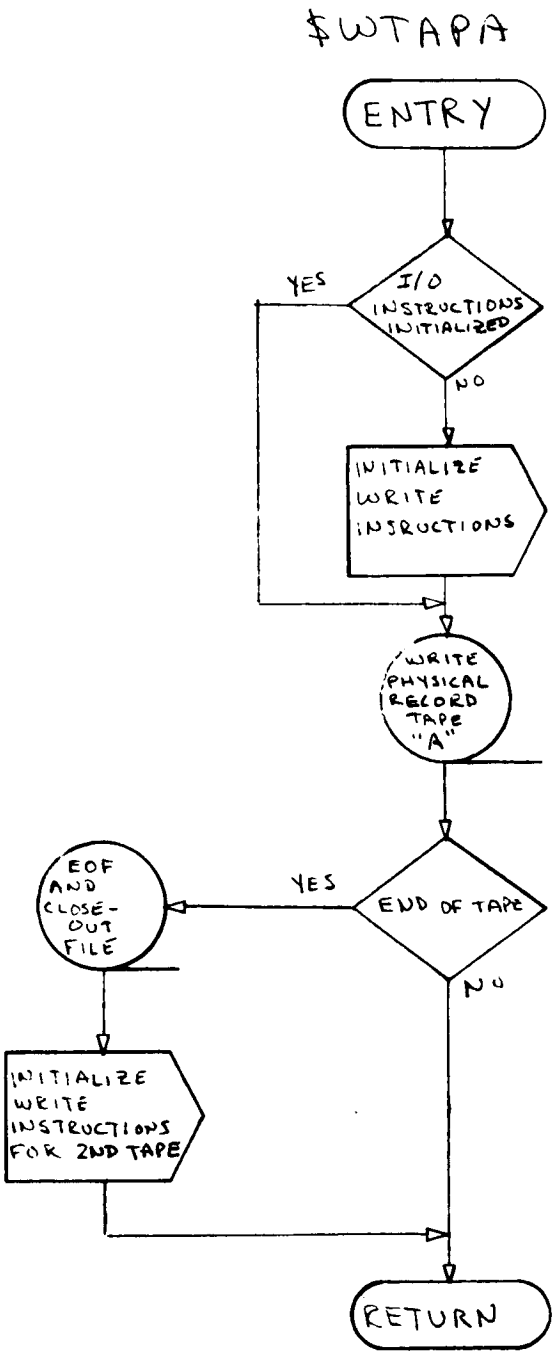
Subroutine WTAPA

Whenever six logical records from $CODER have been stored in an area called BLOCK (6 x 27 = 162 words), WTAPA is entered to write these six on tape A as one physical record.

Upon first entry, all output instructions are set up. After the first entry, WTAPA only writes the 162-word binary record. If one tape is filled, an EOF will be written, followed by a 20-word BCD control record for the SORT subroutine, and another EOF. The tape will be rewound, and the second tape A will immediately be put into use.

After every write, blanks will be stored in every word in the BLOCK area, with the exception of the first three locations of each logical record (BLOCK, BLOCK + 1, BLOCK + 2, BLOCK + 27, BLOCK + 28, etc.); they will be filled with zeroes.

CR-62021

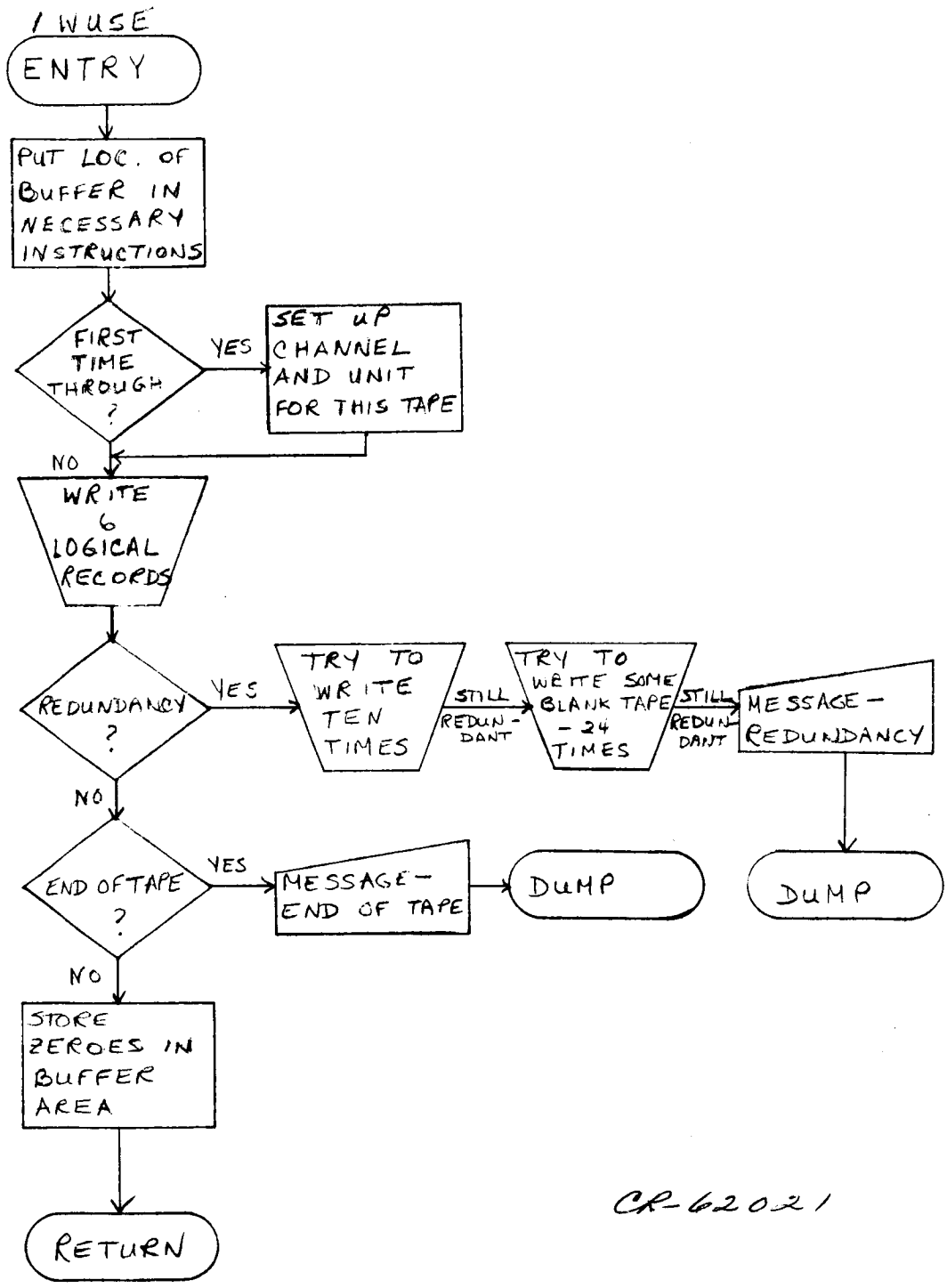

$WTAPA
ENTRY
I/O INSTRUCTIONS INITIALIZED
YES
NO
INITIALIZE WRITE INSRUCTIONS
WRITE PHYSICAL RECORD TAPE "A"
END OF TAPE
YES
NO
EOF AND CLOSE-OUT FILE
INITIALIZE WRITE INSTRUCTIONS FOR 2ND TAPE
RETURN

CR-62021

Subroutine WUSE

This subroutine writes the user tape, which is blocked six logical records per physical record. All processing in this routine is identical to that in the WDOCT subroutine.

CR-62021


/WUSE
ENTRY
PUT LOC. OF BUFFER IN NECESSARY INSTRUCTIONS
FIRST TIME THROUGH ?
YES
SET UP CHANNEL AND UNIT FOR THIS TAPE
NO
WRITE 6 LOGICAL RECORDS
REDUNDANCY ?
YES
TRY TO WRITE TEN TIMES
STILL REDUN-DANT
TRY TO WRITE SOME BLANK TAPE - 24 TIMES
STILL REDUN-DANT
MESSAGE - REDUNDANCY
DUMP
NO
END OF TAPE ?
YES
MESSAGE - END OF TAPE
DUMP
NO
STORE ZEROES IN BUFFER AREA
RETURN

CR-62021

C. SDI MATCH PROGRAM

1. Program Summary

1.1 Description

The SDI Profile Matching Program (MATCH) for the IBM 7090/94 Data Processing System matches coded document profiles against coded user profiles, generating notices when match criteria are met. The program runs under the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with three to five 729 tape units (in addition to FORTRAN units). MATCH consists of a primary program written in FORTRAN and of several subroutines, written in FAP.

In SDI-5, profile descriptor matching is a function of matching single-word descriptors, matching or partially matching phrase descriptors, and, in user profiles, 'must' and 'not' modifiers. Specifically, the MATCH program generates a notice of a document for a user when at least one of the following matching criteria is met:

1. A minimum percentage of matching 'may' (unmodified) single-word descriptors.
2. One matching 'must' descriptor.
3. One matching or partially matching phrase descriptor: two words of a two-word phrase, three of three, three of four, three of five, four of six and five of seven.

But, in any of the above cases, if one matching 'not' descriptor, phrase or single-word is also found, no notice is generated; the 'not' phrase descriptor is subject to the conditions in Item 3 and the 'not' single-word descriptor is equivalent but opposite to a 'must'. Also, in each execution of MATCH, a small percentage of notices may be generated at random to give the user an idea of what he may be missing due to a faulty profile.

The primary MATCH program contains the basic logic of the SDI profile matching phase. It reads and acts on the control card of the program parameters, orders profile input activity, determines whether or not a notice is to be sent and generates notices as required. When the primary program reaches statement 200, the FORTRAN monitor is no longer available; it has been replaced by user profiles in the array USER. This is done to allow as large a buffer as possible for user profiles. The size of array USER is determined by subroutine ERASE, which must be "called" at the end, physically, of the main program.

The important primary program variables that enter the program via control

card are:

1.	NTAPD	FORTRAN logical tape unit, document input
2.	NTAPU	FORTRAN logical tape unit, user input reel 1
3.	NTAPU2	FORTRAN logical tape unit, user input reel 2, if any
4.	NOTIC	FORTRAN logical tape unit, notice output reel 1
5.	NOTIC2	FORTRAN logical tape unit, notice output reel 2, if any
6.	DATE	Date of computer run (optional)
7.	MIN	Minimum match percentage required to generate a notice on the basis of 'may' words
8.	NRAND	Maximum allowable number of random notices per user.

Other important primary program variables are:

1.	NOTES	Total number of notices written
2.	NOTESR	Number of random notices written
3.	KXTRA	1 if normal run, 2 if restart run
4.	KEYGO	Degree of match <0 match on 'not' term (no notice generated) =0 no match at all (no notice generated) 0<KEYGO<100 percentage match on 'may' terms (conditional notice generation) =100 match on single word user 'must' (KEYGO then reset to 300 and a notice generated) 100<KEYGO<300 match on user phrase (notice is generated)
5.	KORE	Length of array USER.

The subroutines used, in order of their logical occurrence within the primary program, are:

1.	ERASE	Calculates the size of array USER and stores the calculated value in cell KORE
2.	DATA	Initializes the reading of document and user profiles from tape
3.	MATCH	Compares the terms of a given pair of user and document profiles in order to determine the number of words found in both profiles i.e., the number of words that match. It returns a four-part answer:

	MATCHS	Number of matching words used as single words
	MATCHP	Number of matching words used in phrases
	N	Number of matching words used both in phrases and as single words
	KEYGO	>0 if match on a 'must' word <0 if match on a 'not' word
4. PHRASE		Attempts to match user phrases with document terms. With a non-zero input argument, it scans for 'not' phrases only. It can reset KEYGO if a 'must' or 'may' phrase matches (100<KEYGO<300) or if a 'not' phrase matches (KEYGO<0).
5. RANDM		Generates random numbers to determine if a random notice is produced at any given time. No user will receive more random notices than specified by the main program parameter NRAND, nor will he receive as random notices any he would (or would not) otherwise have received on the basis of his profile.
6. WRITE		Produces a notice upon command of the primary program.
7. NEXT		Secures a document-user pair of profiles for matching. It automatically controls all profile input after its initialization by subroutine DATA.

1.2 Input

1. Coded document profiles
2. Coded user profiles
3. Control card of program parameters.

1.3 Operating Instructions

The SDI MATCH program is a standard FMS job. The card deck for the system input tape includes in this order: ** job, * ID, *XEQ, source decks if any, binary decks if any, * DATA, control card, EOF. Tape units are assigned via the control card, one or two for user profiles, one or two for notices, and one for document profiles. All tapes must be mounted at the beginning of the job, and switching is then accomplished without operator intervention. A2 is expected as system input. EOJ occurs when matching is concluded or when an EOR is sensed on the

second reel of notice output. Exit is via the standard FMS routines, which are provided for this purpose as part of the MATCH I/O subroutines.

The MATCH program may be restarted at any point after profile matching has begun. To do so, obtain the profile numbers of the pair of profiles being matched (console display or memory dump), or of the last notice generated (last record on notice tape). Punch these numbers into the control card (see format), and begin the run again. The presence of the numbers will cause MATCH to scan to this pair on the user and document tapes, and then to resume matching at the next following pair.

1.4 Output

1. Document Notices

2. Record Formats

2.1 Input

1. Control Card

Cols.	4- 5	FORTRAN logical unit - document input
	9-10	FORTRAN logical unit - user input, reel 1
	14-15	FORTRAN logical unit - user input, reel 2, if any
	19-20	FORTRAN logical unit - notice output, reel 1
	24-25	FORTRAN logical unit - notice output, reel 2, if any
	30-35	Date of computer run
	39-40	Minimum match percentage required on 'may' notices
	44-45	Maximum allowable number of random notices per user
	50-55	User profile number if recovery run, otherwise blank
	60-65	Document profile number if recovery run, otherwise blank.

For document and user profile formats, see the description of the VOCON program.

2.2 Output

1. Document Notices, unsorted, card-image records, 14 words per logical and physical record.

Col.	2	First user initial
	3	Document type
	4	Second user initial
	5	Notice type
		Blank = may
		M = must

CF-62021

P = phrase
9 = random

Cols. 6-15	User surname
16-21	User profile number
23-32	First ten characters of user address
33-38	Document number
40-50	Next eleven characters of user address
77-80	Next four characters of user address

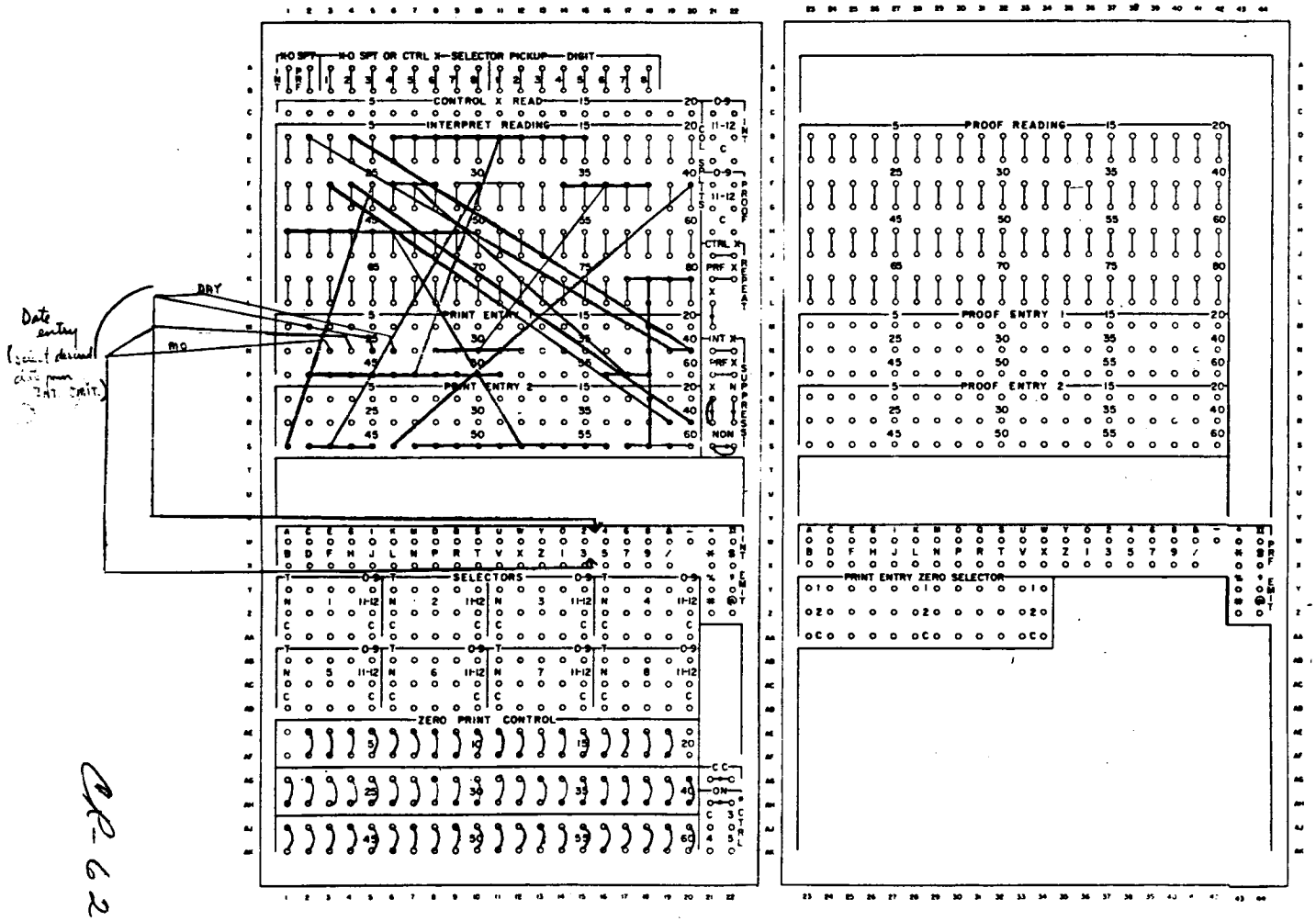
CR 62021

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

Printed in U.S.A.
Form 22-6250-1

557 ALPHABETIC INTERPRETER, CONTROL PANEL DIAGRAM



To be used for interpreting blue NASA cards. To print date, draw month and day from Int. Emit.

To interpret black cards from Black card program (1401), do not include the date. O/P from post cannot be interpreted with this board.

Print entry 1 prints on Line 17

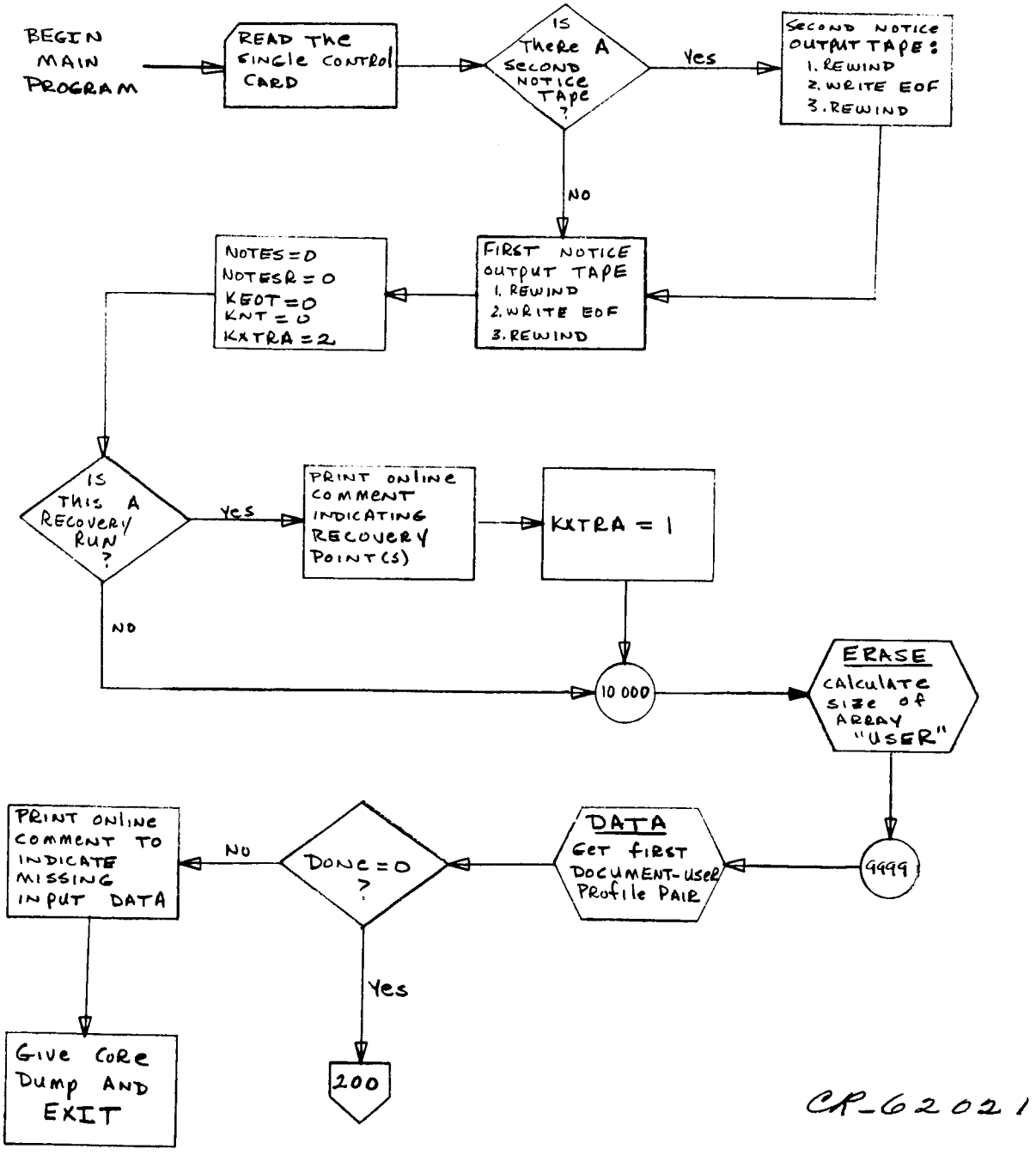
Print entry 2 prints on Line 19

NOTE:

The original of this wiring diagram, with wiring shown in color, was supplied to NASA on September 28, 1964.

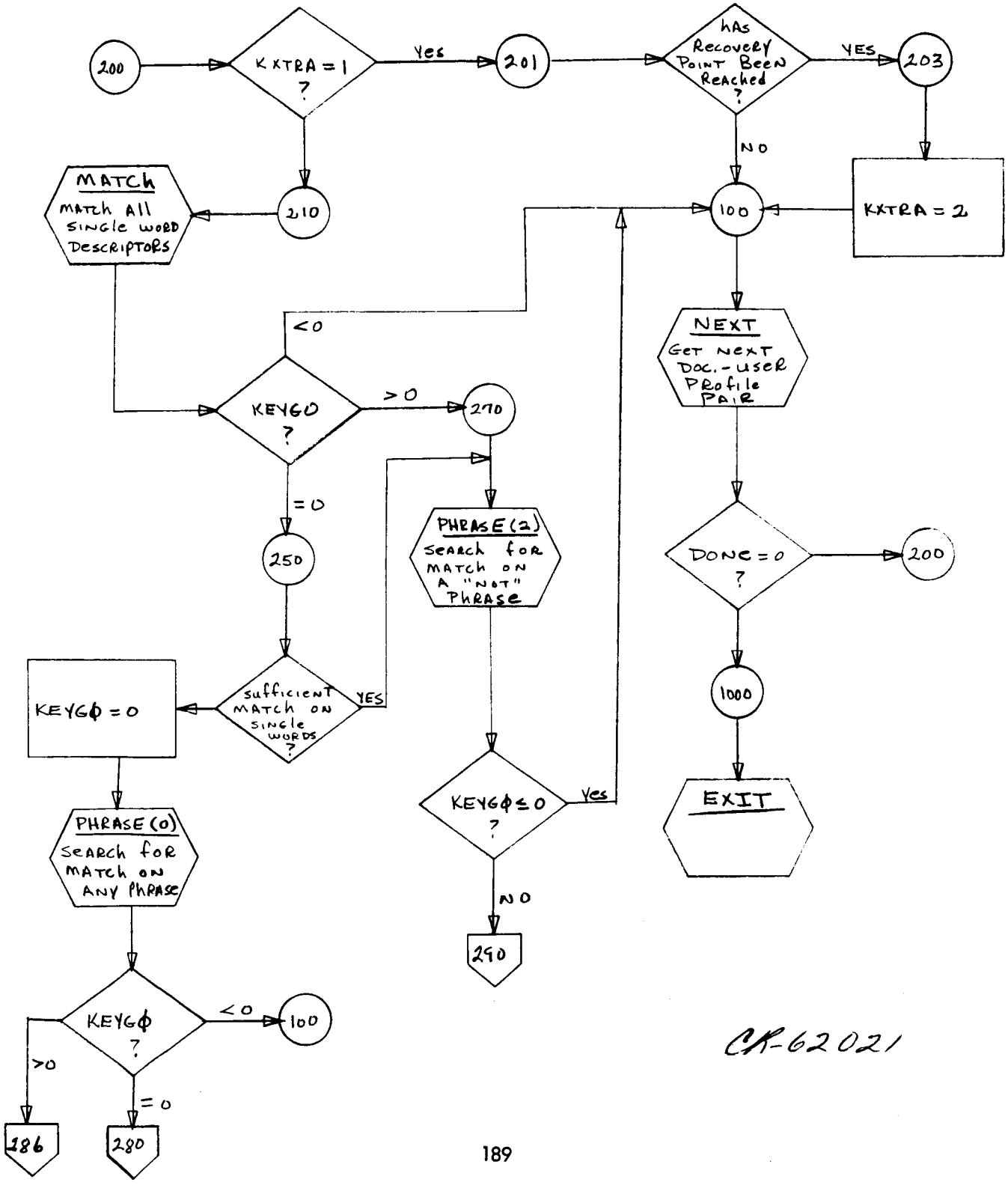
Figure 5. Control Panel Diagram for Interpreting

MAIN MATCH PROGRAM



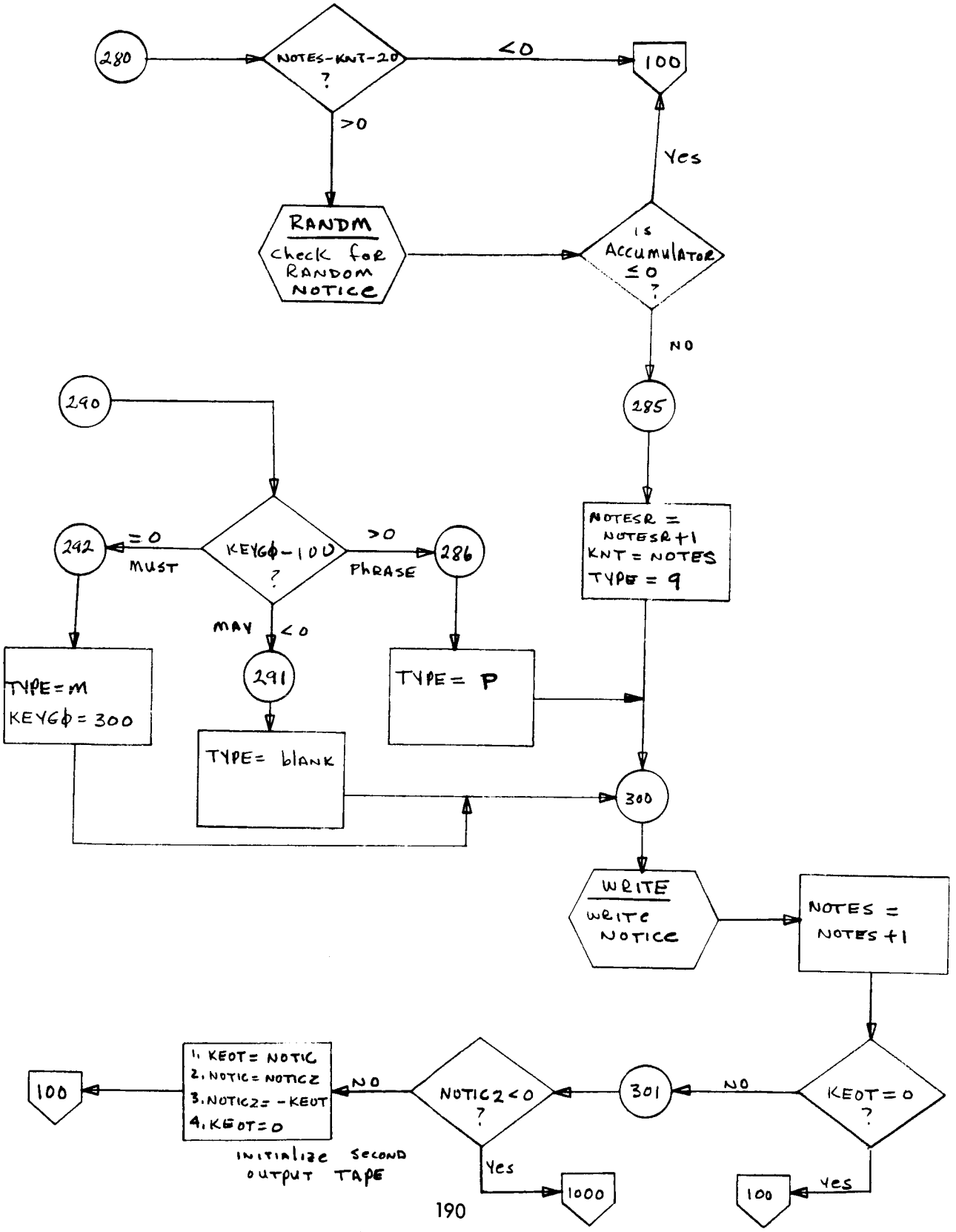
CR-62021

MAIN MATCH PROGRAM

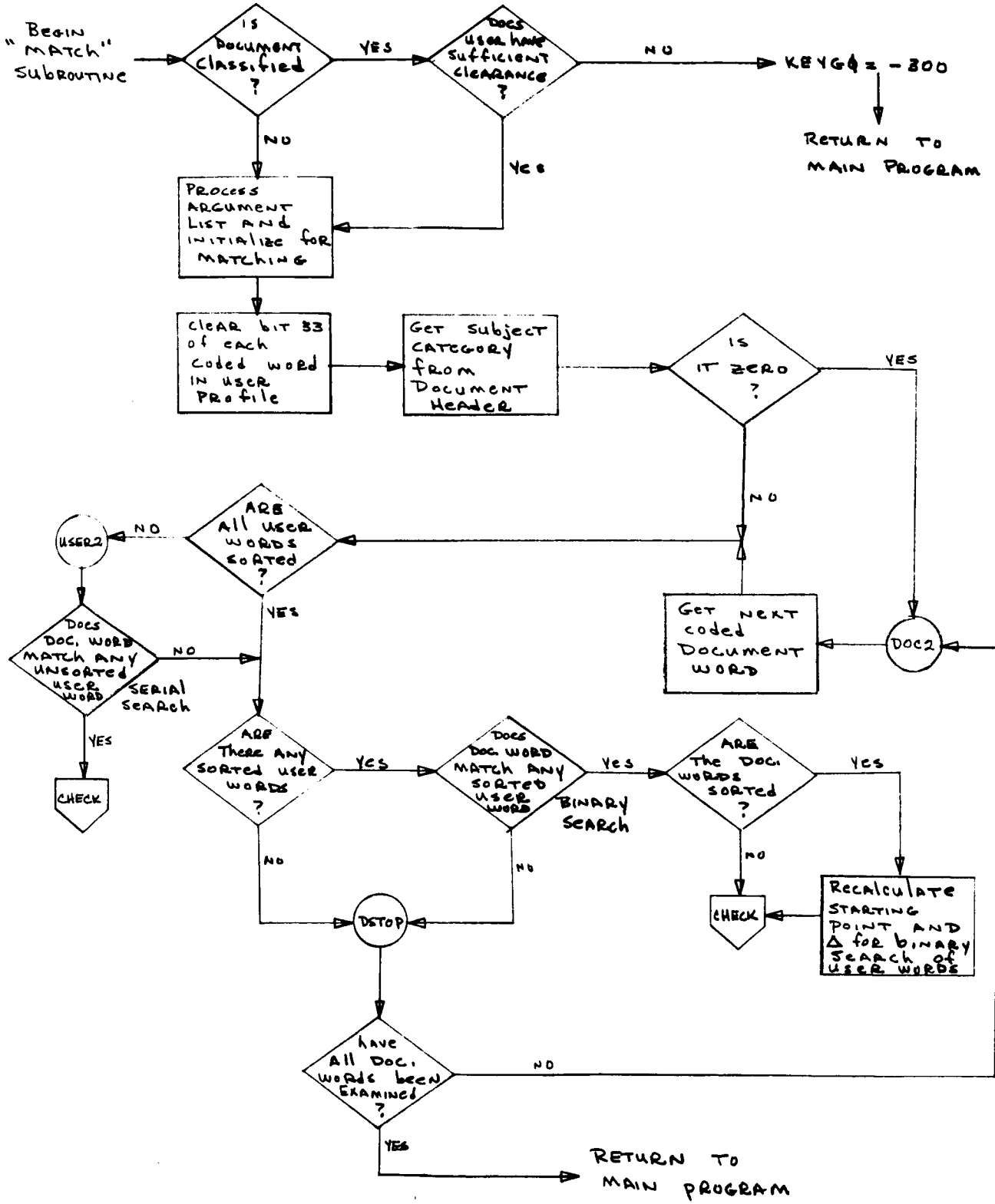


CR-62021

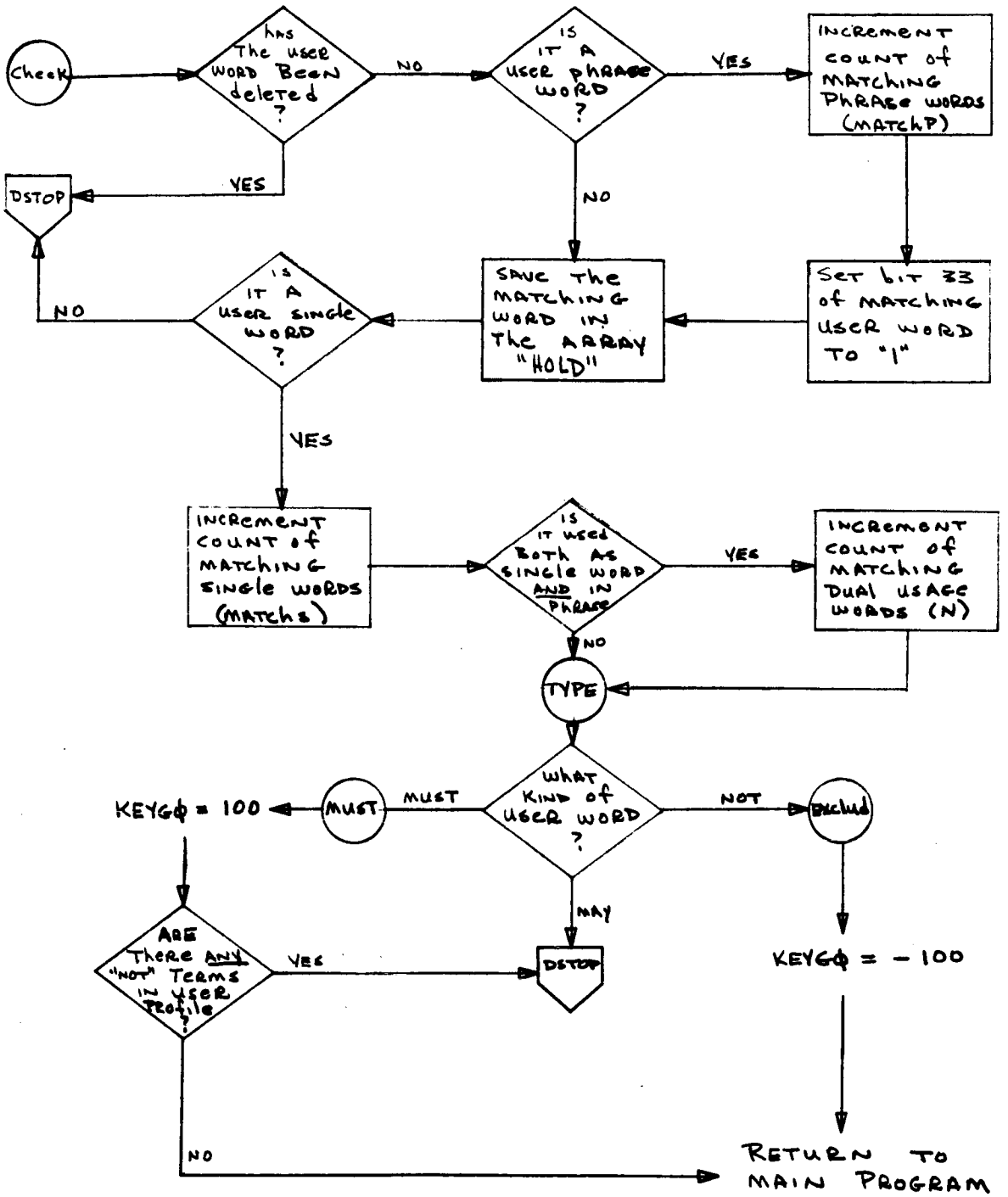
MAIN MATCH PROGRAM



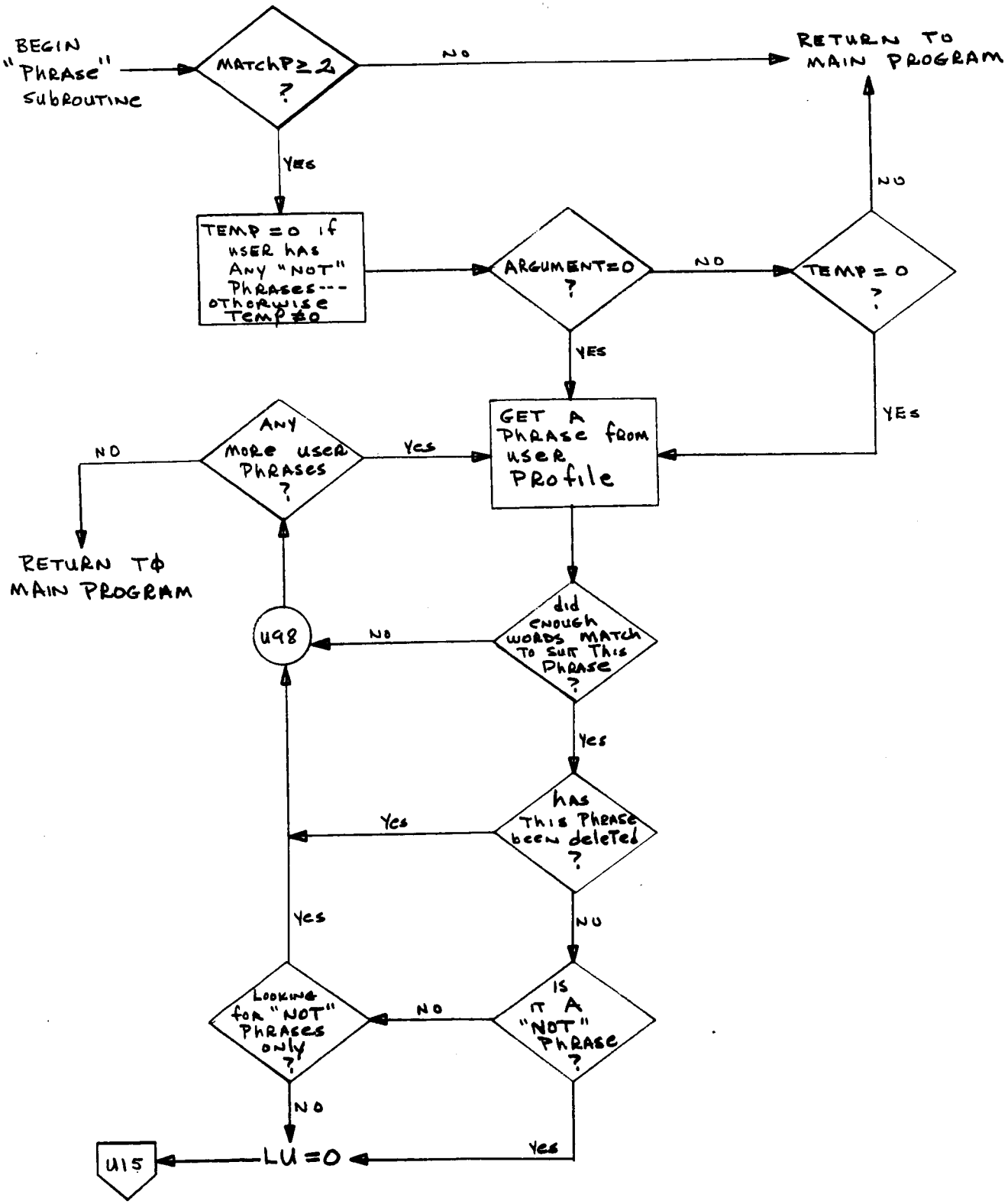
CR-62021


BEGIN "MATCH" SUBROUTINE
IS DOCUMENT CLASSIFIED ?
YES
DOES USER HAVE SUFFICIENT CLEARANCE ?
NO
KEYG@ = -300
RETURN TO MAIN PROGRAM
NO
YES
PROCESS ARGUMENT LIST AND INITIALIZE FOR MATCHING
CLEAR bit 33 of each coded word IN USER Profile
GET SUBJECT CATEGORY FROM DOCUMENT HEADER
IS IT ZERO ?
YES
NO
ARE All USER WORDS SORTED ?
NO
USER2
YES
DOES DOC. WORD MATCH ANY UNSORTED USER WORD
NO
SERIAL SEARCH
YES
CHECK
GET NEXT coded DOCUMENT WORD
DOC2
ARE THERE ANY SORTED USER WORDS ?
YES
DOES DOC. WORD MATCH ANY SORTED USER WORD
YES
BINARY SEARCH
ARE THE DOC. WORDS SORTED ?
YES
NO
NO
NO
CHECK
RECALCULATE STARTING POINT AND Δ for binary SEARCH of USER WORDS
DSTOP
HAVE All DOC. WORDS been EXAMINED ?
NO
YES
RETURN TO MAIN PROGRAM

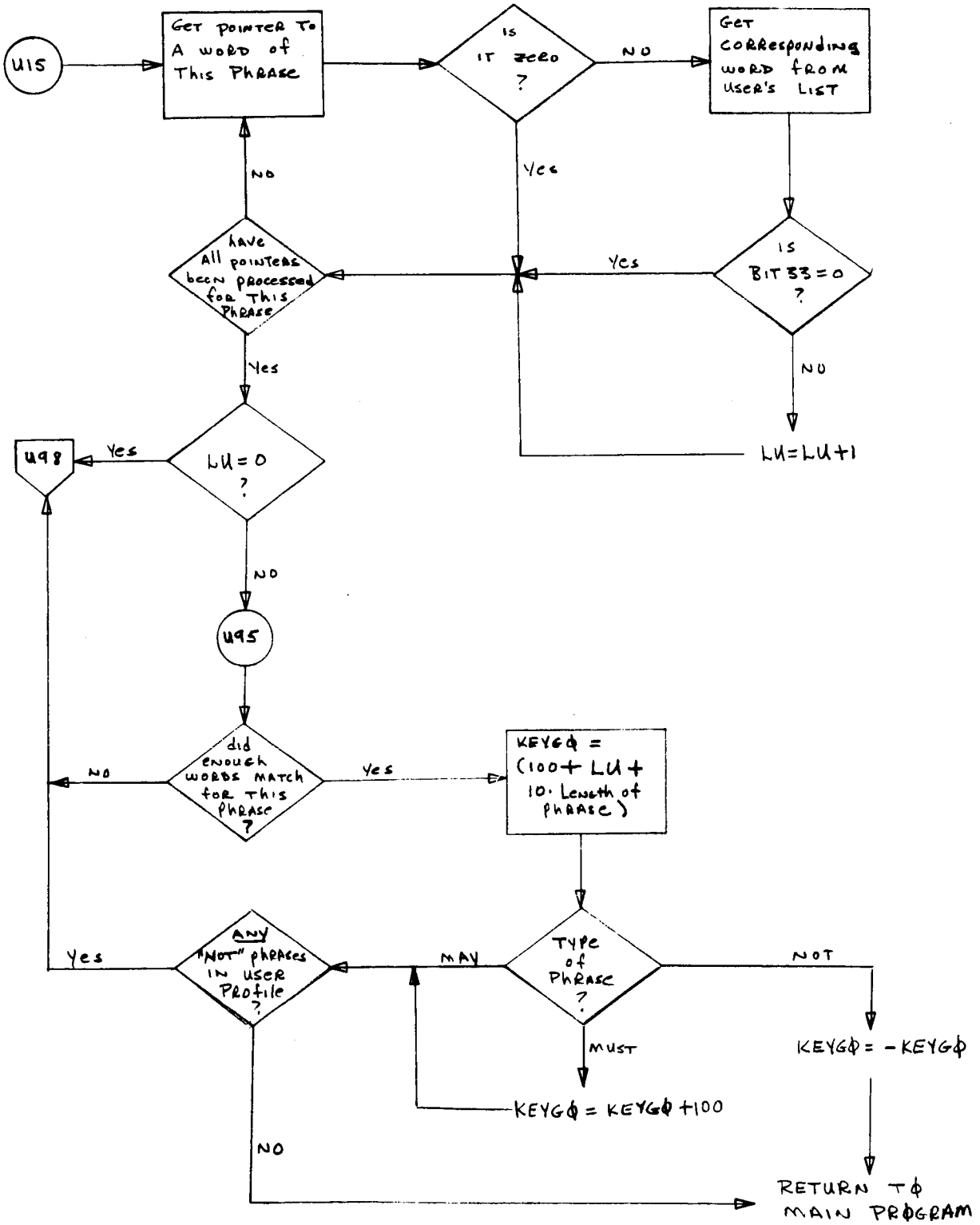
CR-62021


Check
HAS The USER WORD BEEN deleted ?
NO
IS IT A USER PHRASE WORD ?
YES
INCREMENT COUNT of MATCHING PHRASE WORDS (MATCHP)
YES
DSTOP
NO
SAVE the MATCHING WORD IN The ARRAY "HOLD"
Set bit 33 of MATCHING USER WORD TO "1"
IS IT A USER SINGLE WORD ?
NO
YES
INCREMENT COUNT of MATCHING SINGLE WORDS (MATCHS)
IS IT USED BOTH AS SINGLE WORD AND IN PHRASE
YES
INCREMENT COUNT of MATCHING DUAL USAGE WORDS (N)
NO
TYPE
WHAT KIND of USER WORD ?
MUST
MUST
KEYGΦ = 100
NOT
Exclud
MAY
ARE There ANY "NOT" Terms IN USER Profile ?
YES
DSTOP
KEYGΦ = -100
NO
RETURN TO MAIN PROGRAM

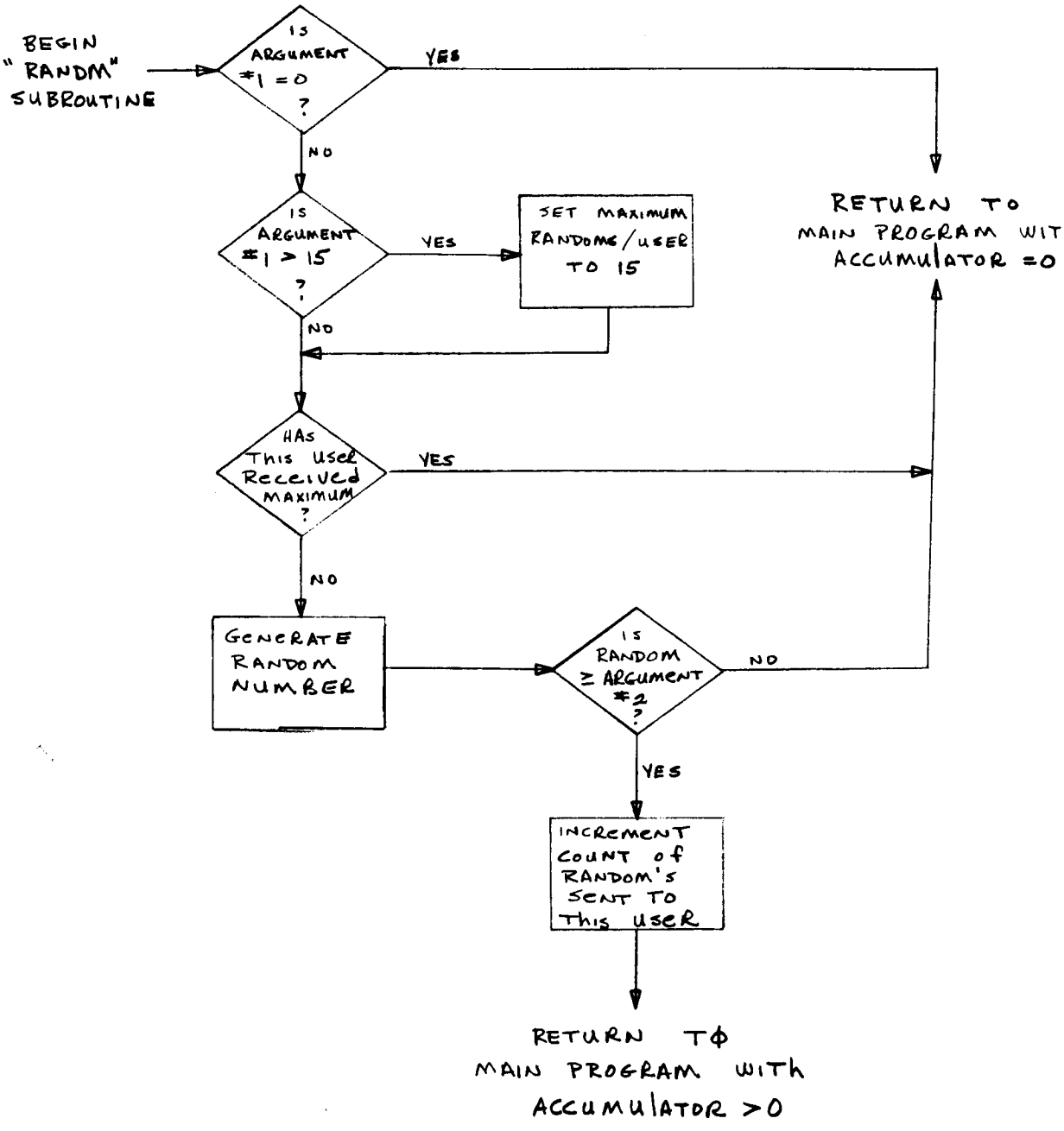
CR-62021


BEGIN "PHRASE" SUBROUTINE
MATCHP≥2 ?
NO
RETURN TO MAIN PROGRAM
YES
TEMP = 0 if USER has ANY "NOT" PHRASES--- OTHERWISE TEMP ≠ 0
ARGUMENT=0 ?
NO
TEMP = 0 ?
NO
YES
YES
GET A PHRASE from USER PROFILE
ANY MORE USER PHRASES ?
NO
YES
RETURN TO MAIN PROGRAM
U98
NO
did ENOUGH WORDS MATCH TO SUIT THIS PHRASE ?
YES
YES
has THIS PHRASE been deleted ?
YES
NO
YES
LOOKING for "NOT" PHRASES only ?
NO
IS IT A "NOT" PHRASE ?
NO
YES
U15
LU=0

CR-62021


U15
GET POINTER TO A WORD of This PHRASE
IS IT ZERO ?
NO
GET CORRESPONDING WORD FROM USER'S LIST
YES
NO
HAVE All POINTERS been PROCESSED for This PHRASE
YES
IS BIT 33 = 0 ?
NO
LU=LU+1
U98
YES
LU = 0 ?
NO
U95
did ENOUGH WORDS MATCH for This PHRASE ?
NO
YES
KEYGØ = (100 + LU + 10 · Length of PHRASE)
ANY "NOT" PHRASES IN USER Profile ?
YES
MAY
TYPE of PHRASE ?
NOT
MUST
KEYGØ = KEYGØ + 100
KEYGØ = - KEYGØ
NO
RETURN TØ MAIN PROGRAM

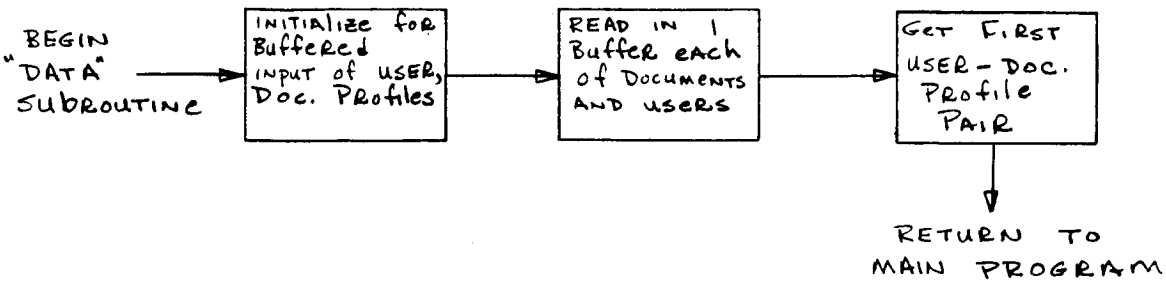
CR-62021

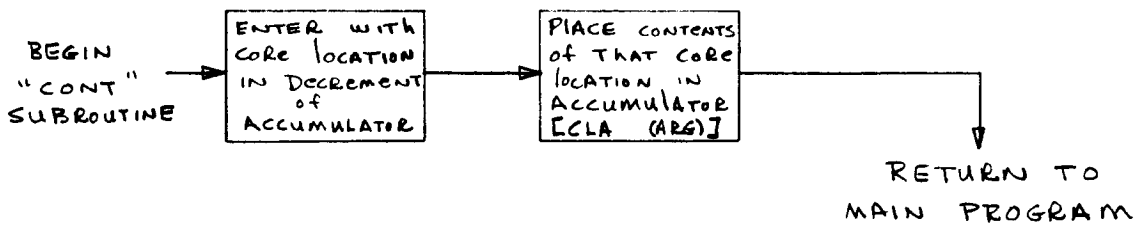

BEGIN "RANDM" SUBROUTINE
IS ARGUMENT #1 = 0 ?
YES
NO
RETURN TO MAIN PROGRAM WIT ACCUMULATOR =0
IS ARGUMENT #1 > 15 ?
YES
SET MAXIMUM RANDOME/USER TO 15
NO
HAS This User Received MAXIMUM ?
YES
NO
GENERATE RANDOM NUMBER
IS RANDOM ≥ ARGUMENT #2 ?
NO
YES
INCREMENT COUNT of RANDOM's SENT TO This USER
RETURN TO MAIN PROGRAM WITH ACCUMULATOR >0

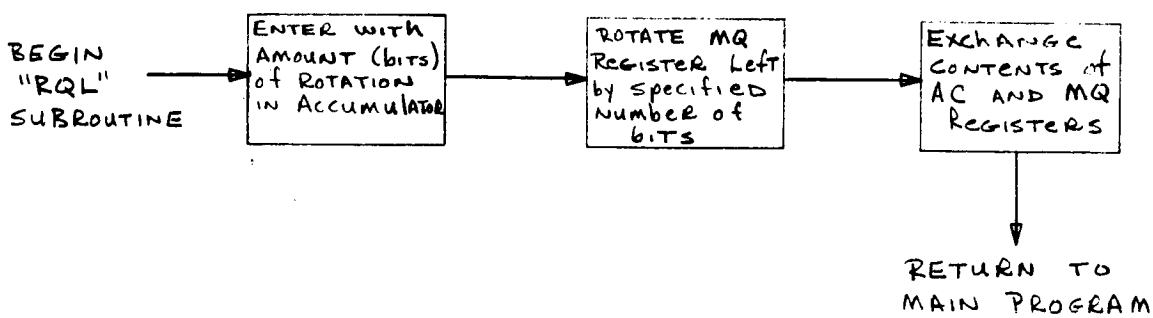
CR-62021

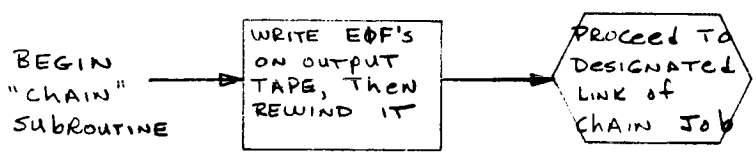
GENERAL IO PACKAGE

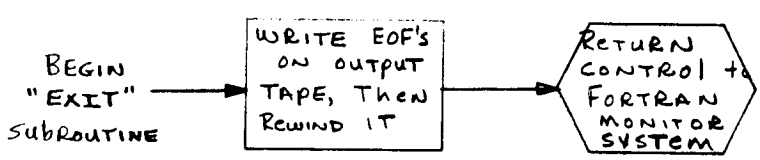
(DATA, CONT, RQL, CHAIN, EXIT)





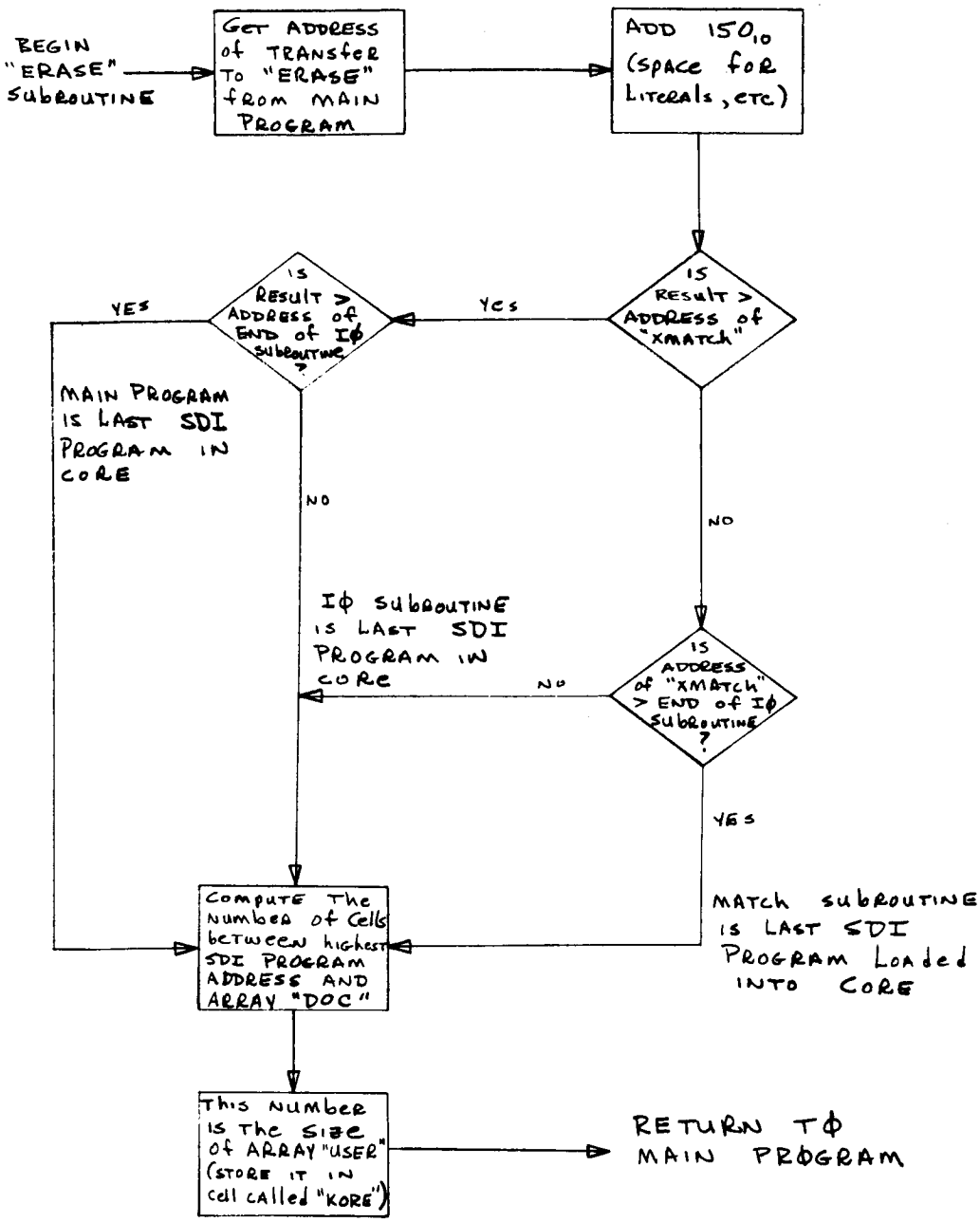






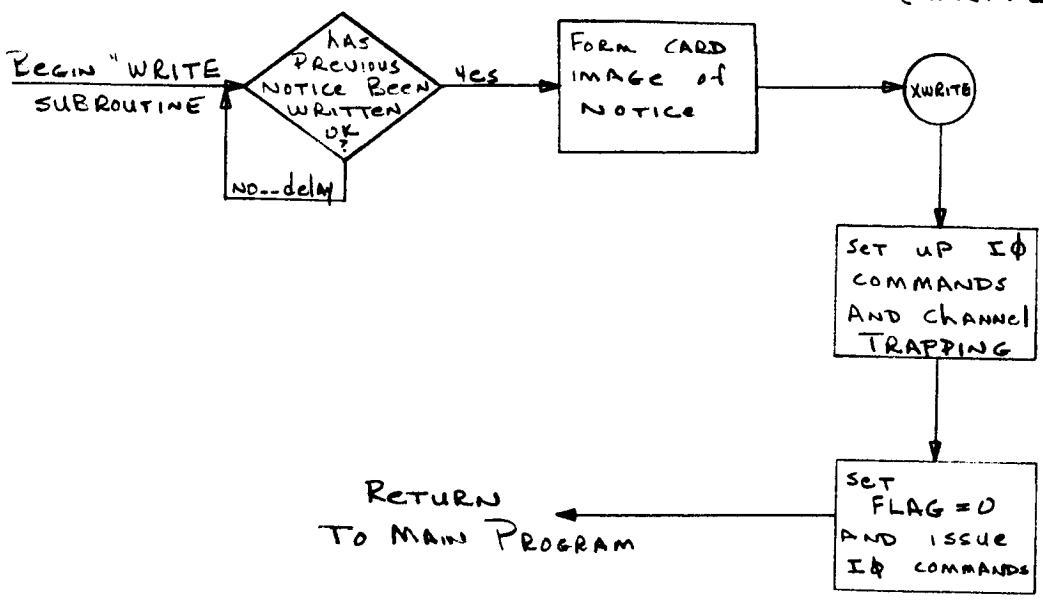
CR-62021

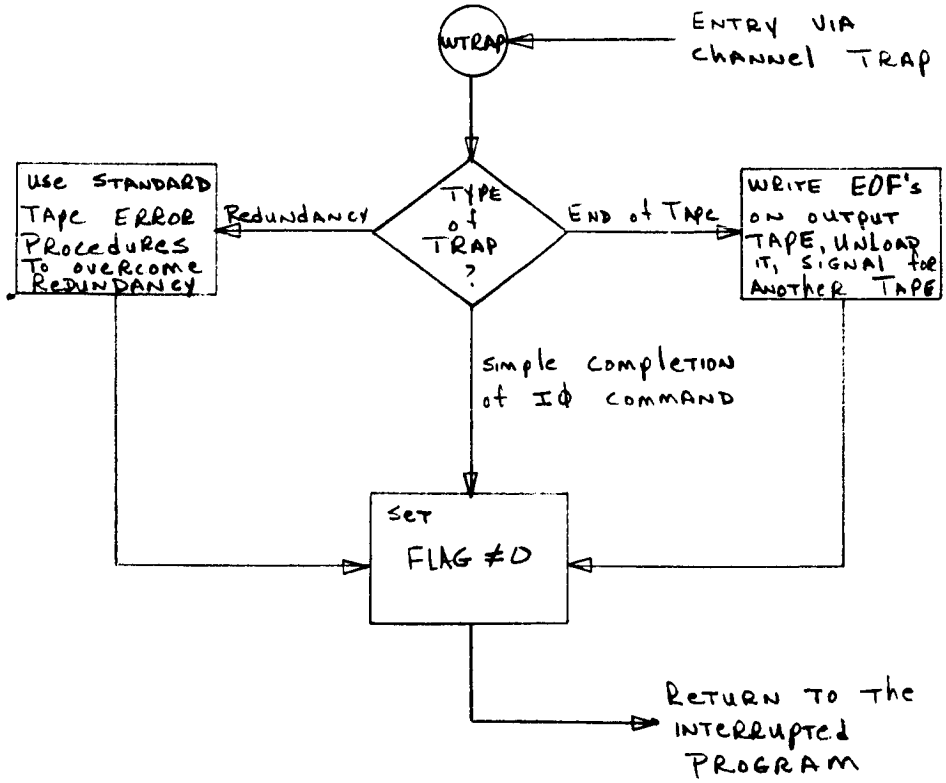
GENERAL IΦ PACKAGE
(ERASE)



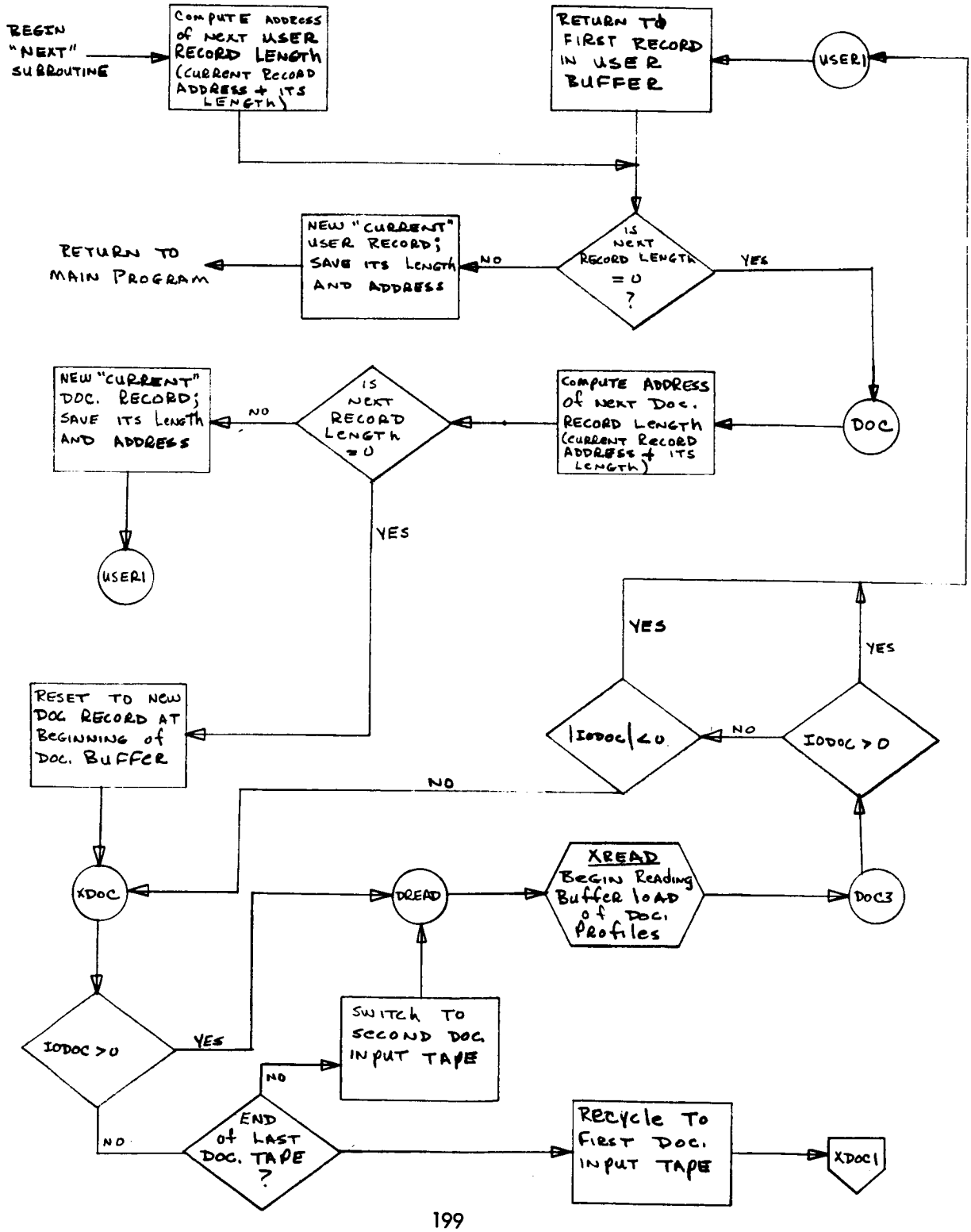
CR-62021

General I-ϕ Package (WRITE)





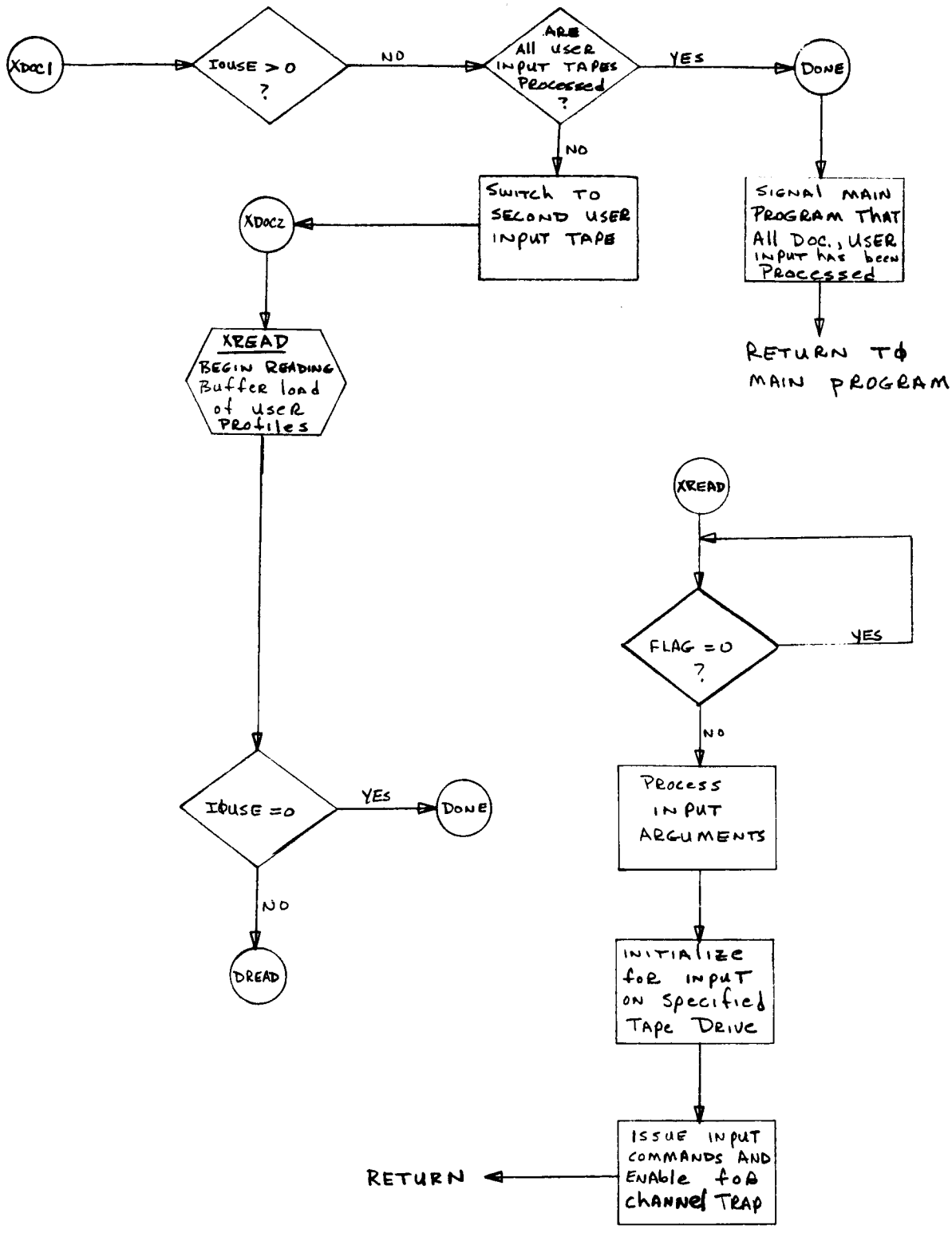
CR-62021


BEGIN "NEXT" SUBROUTINE
COMPUTE ADDRESS of NEXT USER RECORD LENGTH (CURRENT RECORD ADDRESS + ITS LENGTH)
RETURN TO FIRST RECORD IN USER BUFFER
USER1
IS NEXT RECORD LENGTH = 0 ?
NO
YES
NEW "CURRENT" USER RECORD; SAVE ITS LENGTH AND ADDRESS
RETURN TO MAIN PROGRAM
DOC
COMPUTE ADDRESS of NEXT DOC. RECORD LENGTH (CURRENT RECORD ADDRESS + ITS LENGTH)
IS NEXT RECORD LENGTH = 0
NO
YES
NEW "CURRENT" DOC. RECORD; SAVE ITS LENGTH AND ADDRESS
USER1
RESET TO NEW DOC RECORD AT BEGINNING of DOC. BUFFER
|IODOC| < 0
NO
IODOC > 0
YES
YES
NO
XDOC
DREAD
XREAD Begin Reading Buffer load of Doc. Profiles
DOC3
IODOC > 0
YES
SWITCH TO SECOND DOC. INPUT TAPE
NO
END of LAST DOC. TAPE ?
NO
Recycle To FIRST DOC. INPUT TAPE
XDOC1

CR-62021

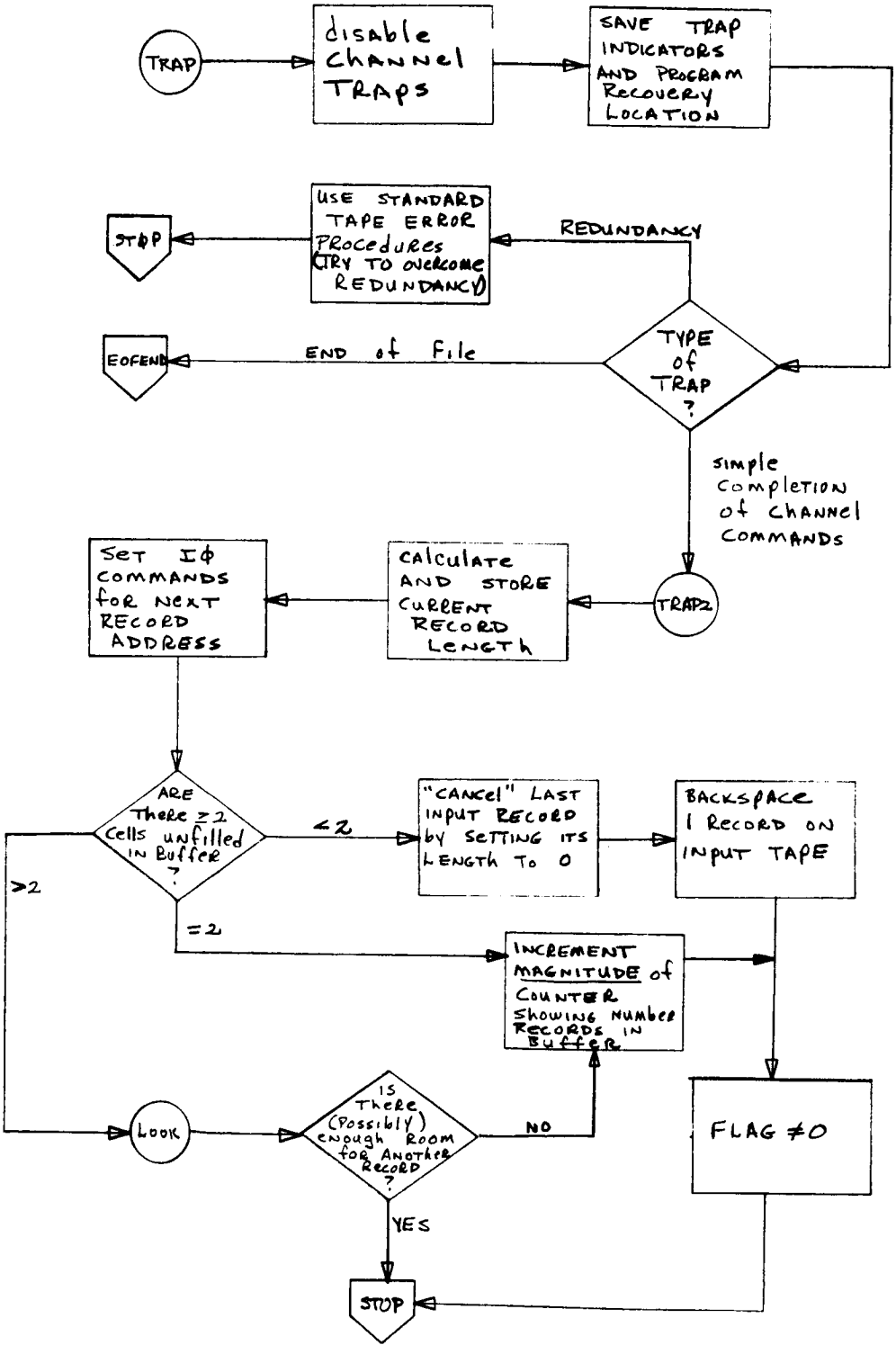
GENERAL IO PACKAGE

(NEXT 2)



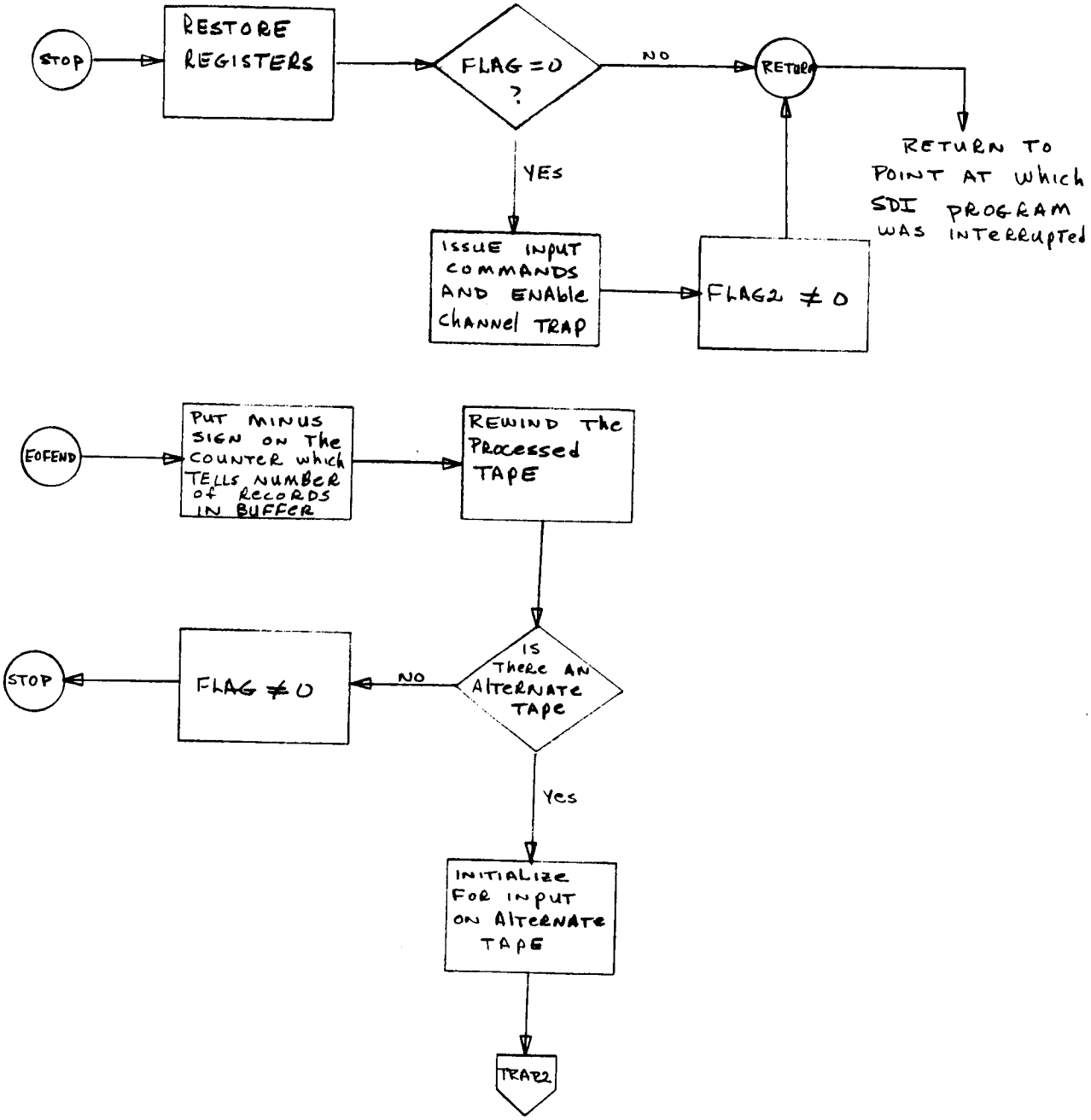
CR-62021

GENERAL IΦ PACKAGE
(Channel Trap – Input)



CF-62021

GENERAL IO PACKAGE
(CHANNEL TRAP - INPUT)



CR-62021

D. SDI POST CARD-TO-TAPE PREPROCESSOR PROGRAM

1. Program Summary

1.1 Description

The SDI Post Card-to-Tape Preprocessor Program (NPOST), for the IBM 1401 Data Processing System, loads on tape user responses to SDI notices and POST program control cards so that the responses may be recorded in the historical notice response file by POST. The program requires a 1402 card read-punch, 1403 printer, two 729 or 7330 tape drives, and, optionally, a 1407 inquiry station; also, 8k core storage, a print line of 132 characters, high-low-equal compare and, optionally, sense switches. NPOST is written in 1401 Autocoder II.

As NPOST loads responses, it checks for validity, counts and analyzes them. It corrects responses where necessary and possible or rejects uncorrectable responses and separates those responses with a comments punch for manual handling. NPOST accepts as many as 6666 responses (the number POST can sort internally). After the last response is proe-essed, NPOST prints the analysis and prepares a backup tape and a response listing.

Since a starting place should be supplied to POST for economical operation, NPOST provides the option either of having the program look for the five lowest document numbers and having the operator choose one of these as the starting place via the inquiry station, or of entering a starting place on a card among the responses.

1.2 Input

1. Card Read Feed: User response cards and POST control cards.
2. Card Punch Feed: Blank cards.

1.3 Operating Instructions

1. Ready the 1401 system: output tapes on units 2 and 3, printer with wide paper, card reader with condensed program deck followed by control and response cards, card punch with blank cards, check stop switch off, sense switch A on, sense switch F on if inquiry station option used.
2. Load program cards.
3. EOJ will be indicated on printer.
4. If sense switch F is on, additional instructions are printed at the inquiry station.

CR-62021

1.4 Output

1. Tape Unit 2: User responses and POST control cards.
2. Tape Unit 3: Duplicate of tape 2.
3. Printer: Analysis and listing of tape 2.
4. Stacker NR: Accepted responses with comments punches.
5. Stacker 1: Accepted responses.
6. Stacker 8/2: Rejected responses with uncorrectable errors.
7. Stacker NP: Blank cards (fed for timing).
8. Card Read Feed: Responses in excess of 6666.

2. Record Formats

2.1 Input

As user responses (Item 7) are returned to the SDI System, each day's accumulation is headed with a date card (Item 6) containing the date of receipt. About four boxes of such date-and-response-card sets are the maximum input to NPOST, headed by POST control cards (Items 1-5). The order of the cards is the order of the following list. For more information on the purposes of the POST control cards, see the description of the POST program.

Supply Items 1-5 as necessary. Only Item 1 is mandatory.

1. Second Notice Date (1 or more, 10 max.)
   Cols. 1 C
   41-44 Date, numeric month and day

2. User Closeout (15 max.), optional.
   Cols. 16-22 User number
   71 7

3. Report Limits (10 max.), optional
   Cols. 1- 6 Lower limit document number
   7-12 Upper limit document number
   71 9

4. POST Starting Place, optional
   Cols. 33-38 Document number
   71 8

5. Document Supply Cancellation (10 max.), optional
   Cols. 1- 6 Lower limit document number
   7-12 Upper limit document number
   71 8

   Supply sets of Items 6 and 7 as explained above.

CF-62021

6. Response Receipt Date
   Cols. 1 * (asterisk)
   41-44 Date, numeric month and day

7. 1st and 2nd Responses (mixed)
   Cols. 16-22 User number
   33-38 Document number
   39 2 if second response
   71,73,75 Response code: +, -, 0, 2 are valid responses, 3 or T in 75 signifies a comment

   Other information punched in the response card is not utilized by NPOST or POST.

2.2 Output

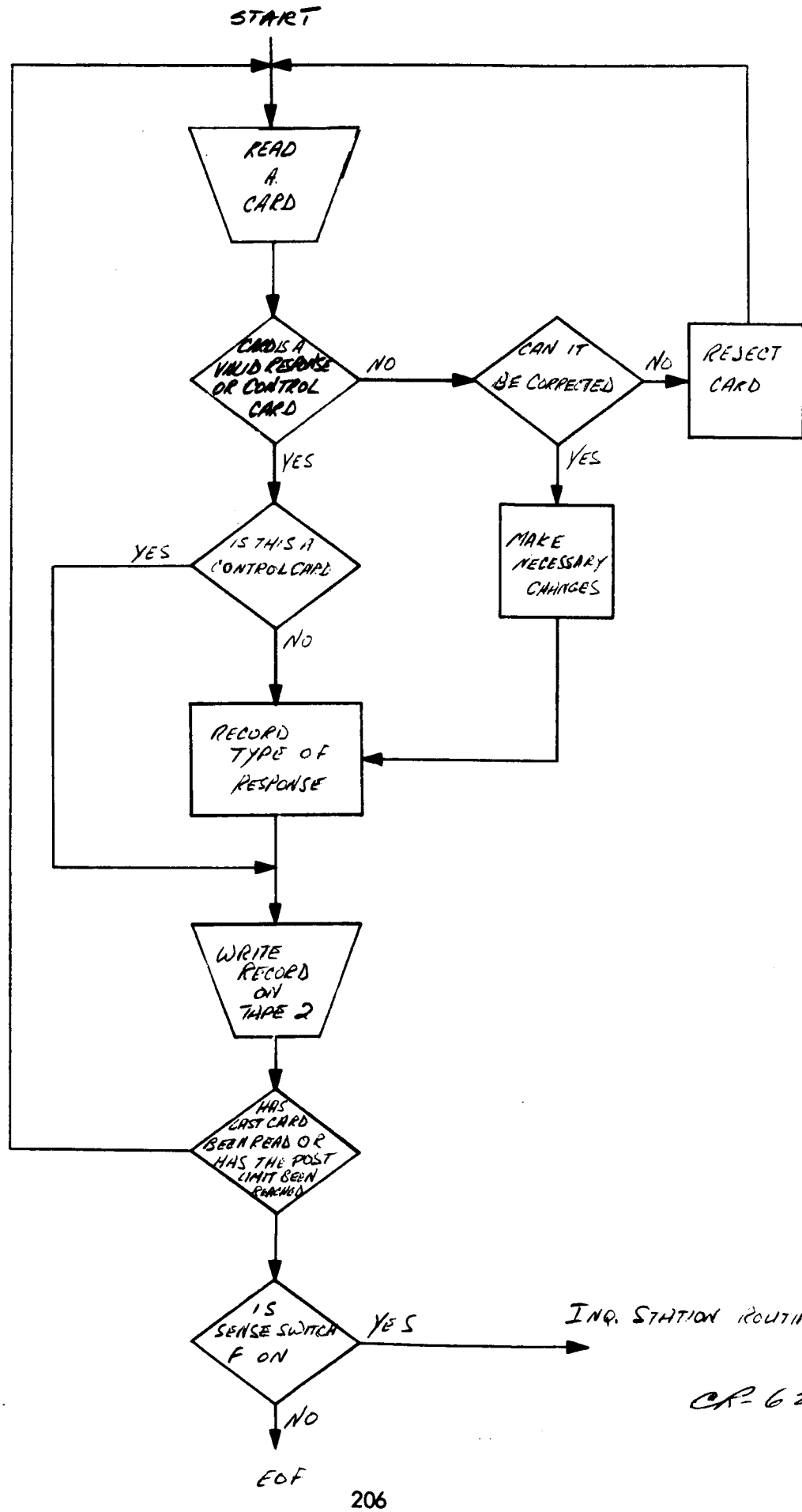
1. Tape Unit 2: Card image of control card and response card input, suitable for BCD input to IBM 7090/94 Data Processing System, 84 characters or 14 words per physical and logical record, one file. Exceptions: column 1 of Items 1 and 6 is blank.

2. Printer: Analysis and listing.

3. Stacker 8/2: Sets of rejected responses, each set preceded by a duplicate receipt date card. The date card is punched whether or not any responses of that set are rejected.

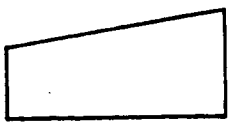
The formats of outputs not listed above are either identical to inputs or are self-explanatory.

CR-62021


START
READ A CARD
CARD IS A VALID RESPONSE OR CONTROL CARD
NO
CAN IT BE CORRECTED
NO
REJECT CARD
YES
YES
IS THIS A CONTROL CARD
YES
MAKE NECESSARY CHANGES
NO
RECORD TYPE OF RESPONSE
WRITE RECORD ON TAPE 2
HAS LAST CARD BEEN READ OR HAS THE POST LIMIT BEEN REACHED
IS SENSE SWITCH F ON
YES
INQ. STATION ROUTINE
NO
EOF

CR-62021

INQUIRY STATION ROUTINE (OPTIONAL)



LIST FIVE (5) LOWEST DOCUMENT NUMBERS

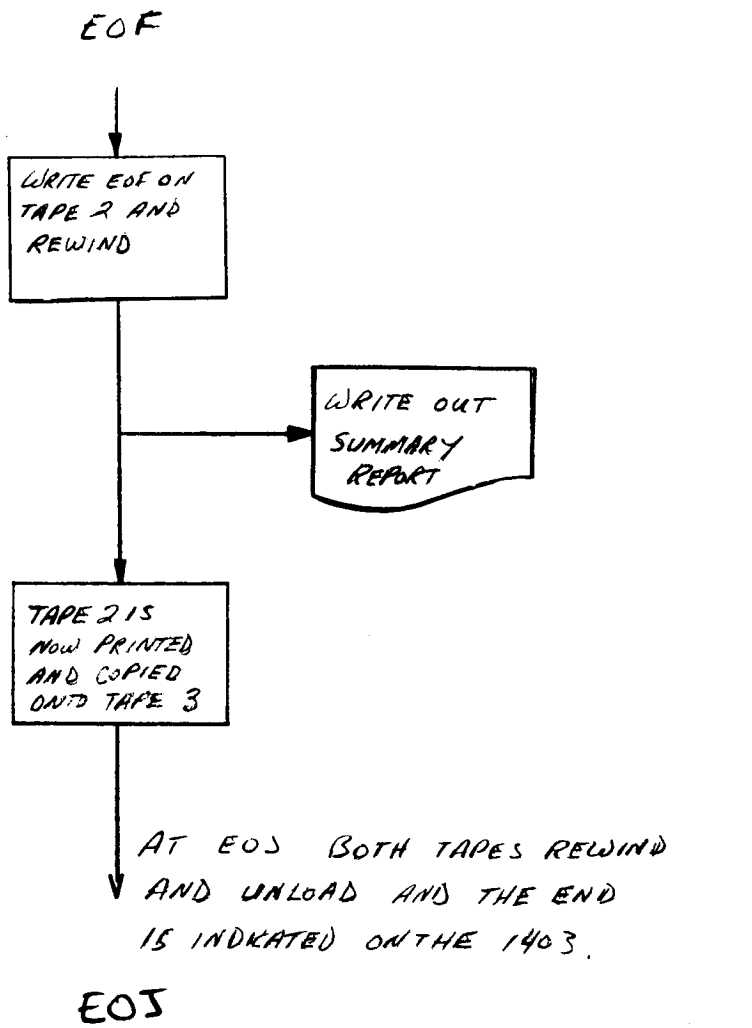
ALLOW OPERATOR TO ENTER STARTING DOCUMENT NUMBER.

IF THE OPERATOR IS SATISFIED WITH HIS CHOICE THEN WRITE THIS DOCUMENT NUMBER ON TAPE FOR POST CONTROL.

BRANCH TO EOF

[If desired the program will indicate the 5 lowest document numbers on the inquiry station and the operator may then indicate where he would want passing to begin. The operating instructions are expressed on the 1407 console printer.]

CR-62021


EOF
WRITE EOF ON TAPE 2 AND REWIND
WRITE OUT SUMMARY REPORT
TAPE 2 IS NOW PRINTED AND COPIED ONTO TAPE 3
AT EOJ BOTH TAPES REWIND AND UNLOAD AND THE END IS INDICATED ON THE 1403.
EOJ

CR-62021

E. SDI POST PROGRAM

1. Program Summary

1.1 Description

The SDI Post Program (POST), for the IBM 7090/94 Data Processing System maintains the historical notice-response file. The program runs under the FORTRAN II Monitor System on a standard IBM 7090 or 7094 computer with 32k core storage and two 7607 data channels with eight 729 tape units (in addition to FORTRAN units). POST is written in FAP.

The historical notice-response file is produced by executions of POST and by the SDI MATCH program. It is maintained in document order. Within a document, notices are ordered by first response, second response and man number. Following the last notice of a given first-response code within a given document is a tally record showing the number of responses with that code, or in other words, the number of notices of the document preceding the tally record. The last tally record for a given document also includes the classification of the document, the current supply of copies and the total notices issued for the document.

The functions of POST in maintaining the historical notice-response file are:

1. Merging notices into the file.
2. Recording or posting user responses to the notices in the file and preparing second notices to accompany documents requested by users.
3. Modifying the file in the cases of users leaving the system and cancellations of document supplies.
4. Printing out portions of the historical file as requested.

The notices to be merged into the historical file are corrected records removed previously because of errors and additions of new material. The responses to be posted to the notices were loaded on tape by NPOST along with POST control cards. In phase 1, POST sorts the responses on document number and saves control information in appropriate tables. Responses for documents below the starting document number (starting-place control card), are dropped to be posted in a later run as are responses for documents not yet added to the historical file. POST then copies the historical file to the starting point, merging notice records as necessary. If any documents passes are within a report-limit range (report-limits control cards), they are written out on the report tape. In phase 2, response posting accompanies the copying and merging of the historical file and alternate notices. All records for a given document - historical notices, notices and responses - are loaded simultaneously in

core. If the document is within a cancellation-limit range (document-supply-cancellation control cards), the supply of copies is set to zero. Then responses and receipt dates are posted (response-receipt-date control cards). For every copy order a dated second notice is prepared if a copy is available (second-notice-date control cards); if not, a memo is prepared. When all responses for that document are posted, the program marks closeouts which are outstanding notices for users who have left the system (user-closeout control cards). Then, it sorts the notices, writes out the document if it is within a report-limits range and writes it out. As each document is processed, statistics are updated to summarize the activity in this run. The final statistics are then written out in a statement at the end of the run.

The tape unit assignments for POST files are set by EQU statements at the beginning of the POST program. They are:

1.	N	Input historical file, channel A, unit 5.
2.	J	Alternate notice file, channel A, unit 6.
3.	M	Response file, channel B, unit 5.
4.	I	Output historical file, channel B, unit 6.
5.	T	Report file, channel A, unit 8.
6.	H	Punch files - second notices, memos, exceptions - channel A, unit 7.
7.	G	Intermediate memo file, channel A, unit 4; copied to Item 6 at EOJ.
8.	F	Intermediate exception file, channel B, unit 1; copied to Item 6 at EOJ.
9.	W	Statement, channel A, unit 3; the system output tape.

Other important POST program variables include:

1.	CHANS	The number of channels used, currently 2.
2.	SIZE	Maximum number of notices per document, currently 1000.
3.	REC	Maximum number of logical records per physical record (blocking) of the historical notice-response file, as output. This variable depends on the setting of sense switch 2 (see Section 1.3).
4.	MAX	MAX = 14 * (REC + 1).
5.	MAXO	MAXO = 14 * (REC).
6.	NRS	Maximum number of responses per execution, currently 6666. The value of (3 * NRS) cannot exceed the value of (20 * SIZE). POST performs an internal sort on the responses; 6666 is the

CR-62021

maximum number that can be sorted.

7. CSIZE — Maximum number of closeouts per execution, currently 15.
8. RSIZE — Maximum number of report intervals per execution, currently 10.
9. SSIZE — Maximum number of supply cancellation intervals per execution, currently 10.
10. RQSIZE — Number of second notices to receive the same date, currently 250. RQSIZE is a reflection of the number of copies that the SDI clerks can mail in one day.
11. LPATCH — The address of the first unused location in the area PATCH. Because POST uses all space above itself for data, an area called PATCH is reserved for changes. LPATCH must be updated as the program is patched.

1.2 Input

1. Historical notice-response file.
2. Alternate notice file (for merging).
3. Responses and control cards (from NPOST).

1.3 Operating Instructions

The SDI POST program is a standard FMS job. The card deck for the system includes in this order: ** job, * ID, * XEQ, source decks if any, binary decks if any, EOF. A2 is expected as system input; A3 as system output; POST requires the I/O table from VOCON.

Sense switch settings are:

1. SS2 — ON if historical notice-response file is to be written unblocked.
2. SS5 — ON if alternate notice file is present.
3. SS1 — Set ON when last reel of historical file is mounted; on throughout run if only one reel.
4. SS3 — Set ON when last reel of alternate notice file is mounted; on throughout run if only one reel; OFF if no alternate file.

Program stops are:

1. $(3363)_8$ — If EOR historical file input, mount next reel and press START.
   If EOF, set on SS1 and press START.

CR 62021

2. $(3372)_8$ If EOR alternate notice file, mount next reel and press START.
If EOF, set on SS3 and press START.

3. $(3530)_8$ EOT on output historical notices; mount next reel and press START.

4. $(2631)_8$ EOJ. POST destroys any part of the monitor above itself, and thus cannot exit through the monitor.

1.4 Output

1. Historical notice-response file (updated)
2. Second notices
3. Memos
4. Exceptions
5. Document reports
6. Statement.

2. Record Formats

2.1 Input

1. Historical Notice-Response File

   Starred fields are added to the records by the POST program. The file is sorted on document number, first response, second response and man number. It may be blocked or unblocked; see variable REC.

1a. Notice Records

Char.	
2	First user initial
4	Second user initial
6-15	User surname
16-21	User profile number
33-36	*Date of first notice
37-40	*First response receipt date
41-44	*Date of second notice
45-48	*Second response receipt date
55-60	Document number
71	*Response code of first response
72	*Response code of second response
75	*Digit 3 if comments punch
80	Notice type
	Blank = may
	M = must
	P = phrase
	9 = random

CR-62021

The allowable response codes in ascending sort order are:

+	Of interest, copy requested
-	Of interest, copy not requested
0	Of interest, have seen before
2	Of no interest
7	User closed out without response
Blank	No response

1b. Tally Records

POST expects a final tally record for each document from the MATCH program; if one is not provided, the document is assumed unclassified and the copy supply unlimited. POST creates intermediate tally records containing the fields identified by +.

Char.	
1	Classification Blank = unclassified + = classified
+50-53	*Number of notices with given first-response code
+55-60	Document number
64-65	Copy supply
74-77	Total number of notices for the document.

2. Alternate Notice File

The format of this file is the same as that of Item 1, unblocked.

3. Responses and Control Cards

The formats of these cards are listed in the discussion of NPOST, the program that loads them on tape.

2.2 Output

1. Historical Notice-Response File (updated)

The format of this file is the same as that of Item 1, input.

2. Second Notices

The format of this file is the same as the format of the second response, listed in the discussion of NPOST, with the exception that the response field has not yet been punched by the user.

3. Memos

The format of this file is the same as the format of the second response, listed in the discussion of NPOST, with the exception of the response field and the second-response identification code.

4. Exceptions

This file includes all valid responses that do not fall within the document range processed by a given execution and any notices

CR-62021

rejected as being out of document order. These records have the formats of their respective files, Items 2 and 1, input.

5. Document Reports

   The format of this file is the same as Item 1, input, unblocked.

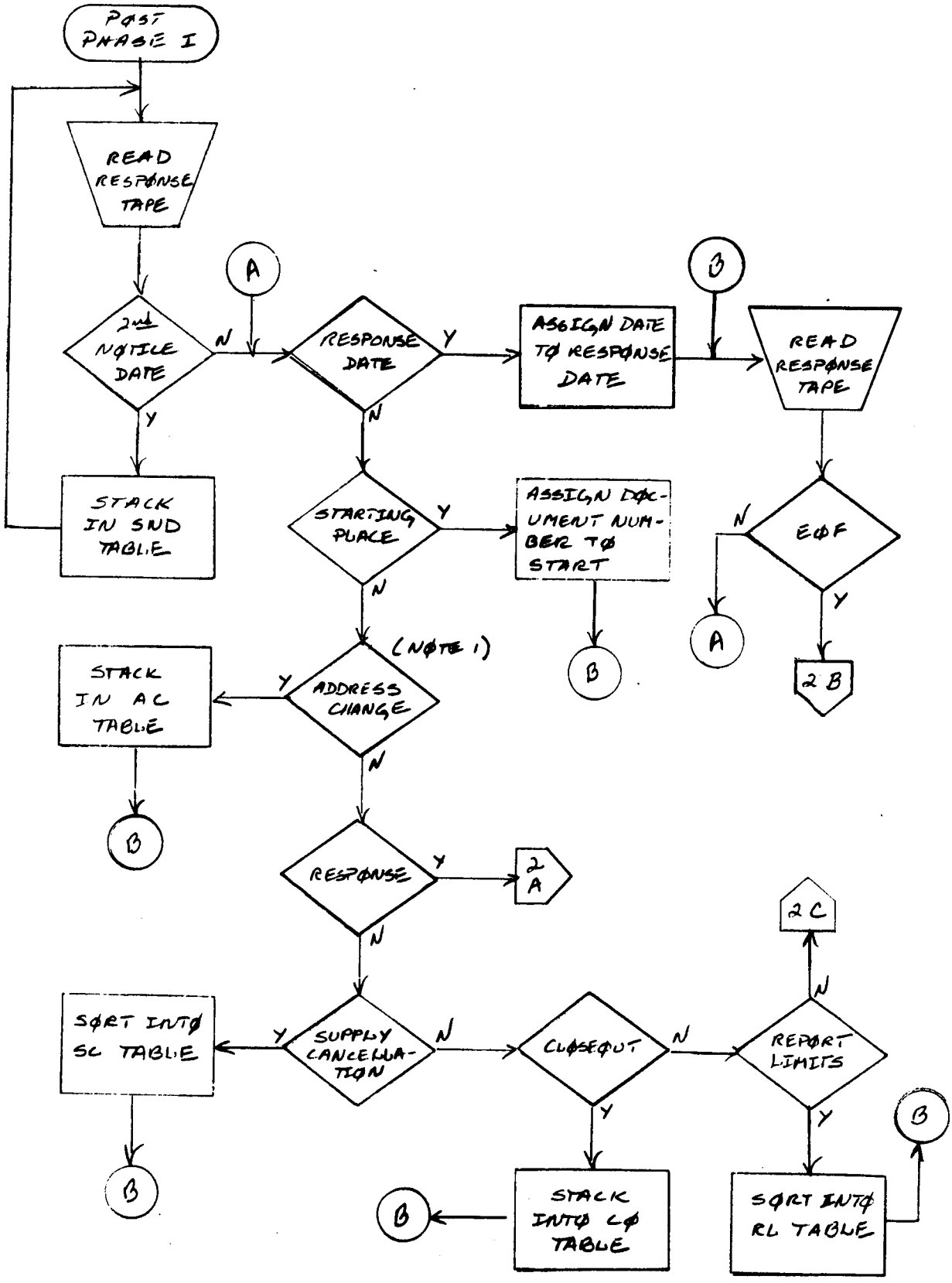
6. Statement

CR-62021

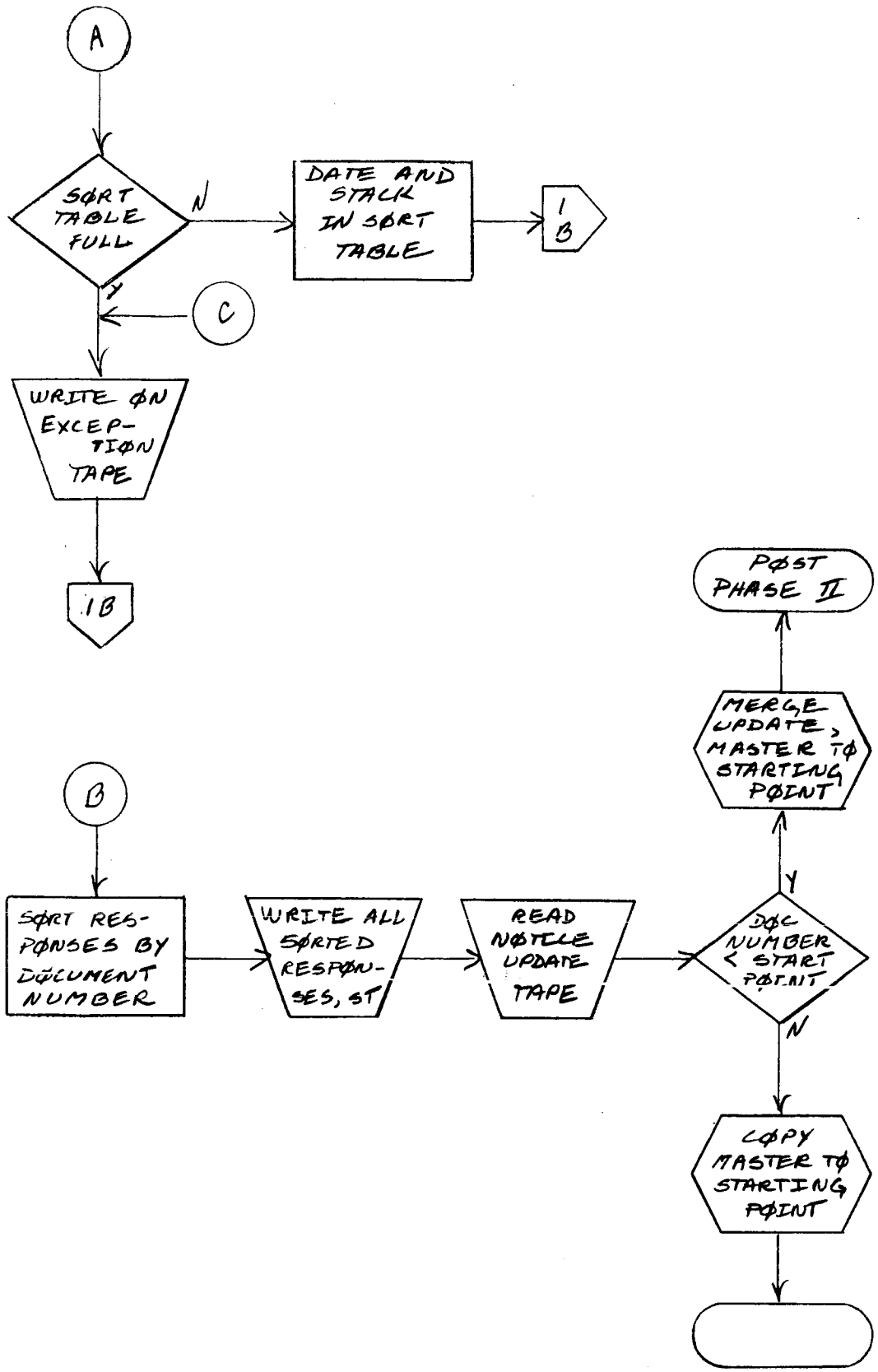
NOTE 1

The POST program was developed in another context. It is a useful adjunct to the NASA-SDI system, but as submitted, is not integrated completely into the system.

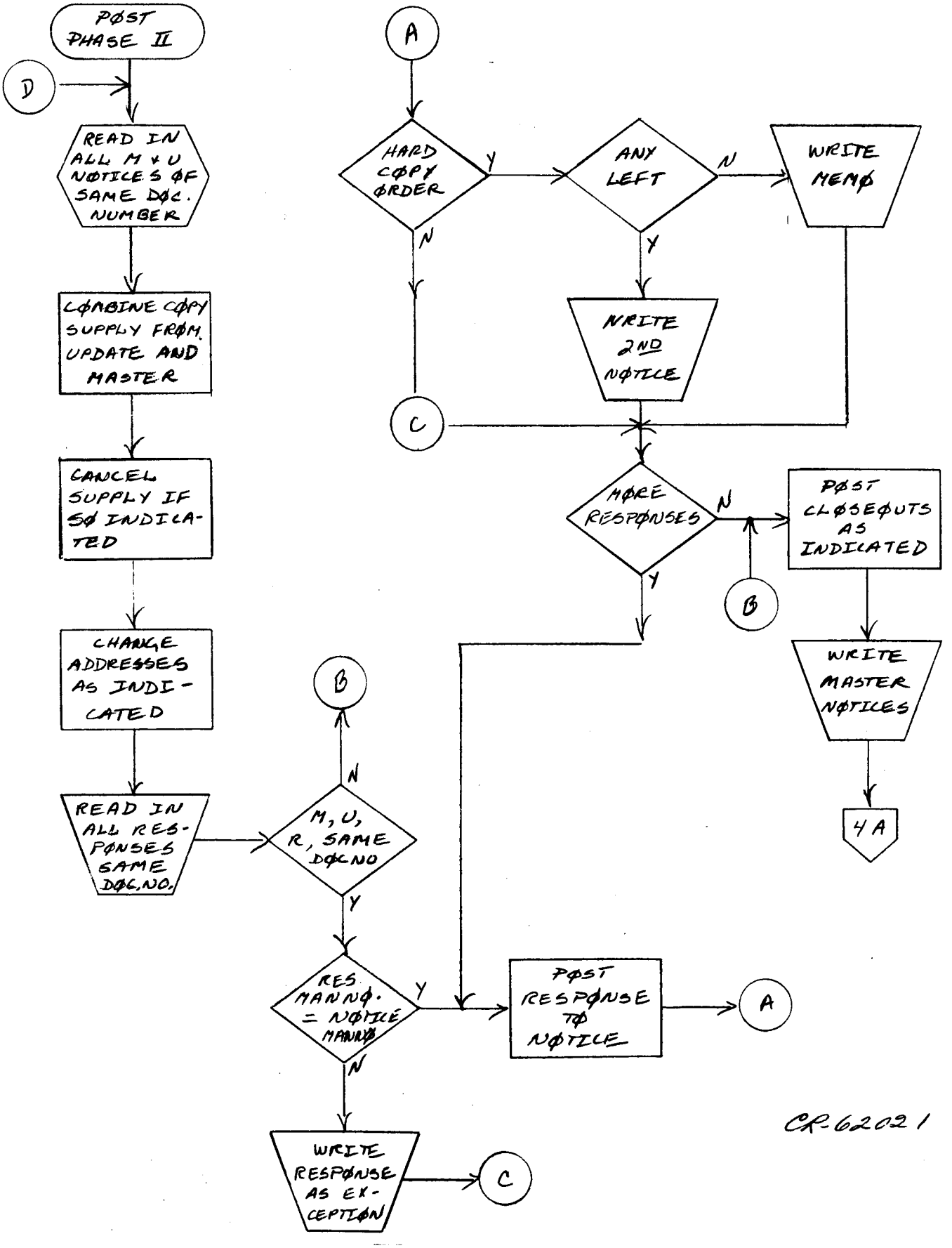
CR-62021


POST PHASE I
READ RESPONSE TAPE
2nd NOTICE DATE
N
Y
A
RESPONSE DATE
Y
N
ASSIGN DATE TO RESPONSE DATE
B
READ RESPONSE TAPE
STACK IN SND TABLE
STARTING PLACE
Y
N
ASSIGN DOCUMENT NUMBER TO START
B
EOF
N
Y
A
2 B
(NOTE 1)
STACK IN AC TABLE
Y
ADDRESS CHANGE
N
B
RESPONSE
Y
N
2 A
2 C
SORT INTO SC TABLE
Y
SUPPLY CANCELLATION
N
CLOSEOUT
N
REPORT LIMITS
N
Y
Y
B
B
B
STACK INTO CO TABLE
SORT INTO RL TABLE

CF-62021


A
SORT TABLE FULL
N
DATE AND STACK IN SORT TABLE
1 B
Y
C
WRITE ON EXCEP-TION TAPE
1B
POST PHASE II
MERGE UPDATE, MASTER TO STARTING POINT
B
SORT RES-PONSES BY DOCUMENT NUMBER
WRITE ALL SORTED RESPON-SES, ST
READ NOTICE UPDATE TAPE
DOC NUMBER < START POINT
Y
N
COPY MASTER TO STARTING POINT

CR-62021


POST PHASE II
D
READ IN ALL M + U NOTICES OF SAME DOC. NUMBER
COMBINE COPY SUPPLY FROM UPDATE AND MASTER
CANCEL SUPPLY IF SO INDICATED
CHANGE ADDRESSES AS INDICATED
READ IN ALL RESPONSES SAME DOC,NO.
M, U, R, SAME DOC NO
N
B
Y
RES. MAN NO. = NOTICE MAN NO
Y
N
WRITE RESPONSE AS EXCEPTION
C
POST RESPONSE TO NOTICE
A
A
HARD COPY ORDER
Y
ANY LEFT
N
WRITE MEMO
N
Y
WRITE 2ND NOTICE
C
MORE RESPONSES
N
Y
B
POST CLOSEOUTS AS INDICATED
WRITE MASTER NOTICES
4 A

CR-62021

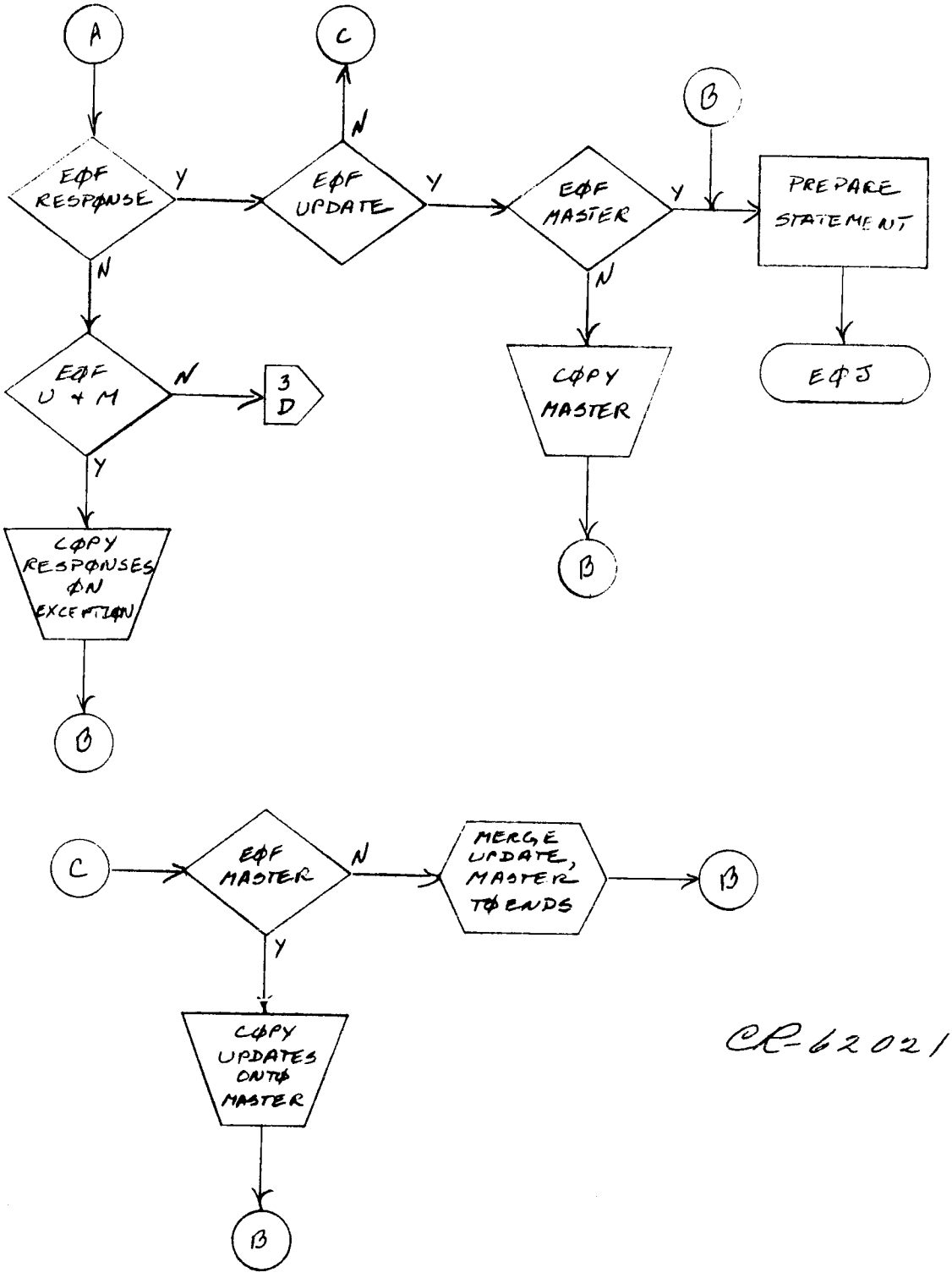
NASA-SDI SYSTEM

CONTENTS:

This volume is solely devoted to the documentation of the programs for the NASA-SDI System.

CR-62021

Page vi blank


A
C
B
EOF RESPONSE
Y
N
EOF UPDATE
Y
N
EOF MASTER
Y
N
PREPARE STATEMENT
EOJ
EOF U + M
N
3 D
Y
COPY RESPONSES ON EXCEPTION
COPY MASTER
B
B
C
EOF MASTER
N
Y
MERGE UPDATE, MASTER TO ENDS
B
COPY UPDATES ONTO MASTER
B

CR-62021